Vysoká škola báňská – Technická univerzita Ostrava

Ekonomická fakulta

KATEDRA APLIKOVANÉ INFORMATIKY

Návrh a implementace prodejního a skladového systému

Design and Implementation of Selling and Warehouse Systém

Student: Břetislav Glac

Vedoucí bakalářské práce: RNDr. Ivo Martiník, Ph.D.

Ostrava 2013

VŠB - Technická univerzita Ostrava
Ekonomická fakulta
Katedra aplikované informatiky

Zadání bakalářské práce

Student: **Břetislav Glac**

Studijní program: B6209 Systémové inženýrství a informatika

Studijní obor: 6209R001 Aplikovaná informatika

Téma: Návrh a implementace prodejního a skladového systému
Design and Implementation of Selling and Warehouse System

Zásady pro vypracování:

1. Úvod
2. Informační systémy – vlastnosti, typy, metody vývoje
3. Analýza nabídky konkurenčních systémů a aplikací
4. Návrh a implementace systému
5. Závěr

Seznam použité literatury
Seznam zkratek
Prohlášení o využití výsledků bakalářské práce
Seznam příloh
Přílohy

Seznam doporučené odborné literatury:

BRUCKNER, Tomáš. *Tvorba informačních systémů: principy, metodiky, architektury.* Praha: Grada, 2012. ISBN 978-80-247-4153-6.
BASL, Josef a Roman BLAŽÍČEK. *Podnikové informační systémy: Podnik v informační společnosti.* Praha: Grada, 2012. ISBN 978-80-247-4307-3.
ČADA, Ondřej. *Objektové programování: naučte se pravidla objektového myšlení.* Praha: Grada, 2009. ISBN 978-80-247-2745-5.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **RNDr. Ivo Martiník, Ph.D.**

Datum zadání: 23.11.2012

Datum odevzdání: 10.05.2013

Ing. Petr Rozehnal, Ph.D.
vedoucí katedry

VYSOKÁ ŠKOLA BÁŇS. - TECHNICKÁ UNIVERZITA Ekonomická fakulta 1 OSTRAVA

prof. Dr. Ing. Dana Dluhošová
děkanka fakulty

Místopřísežné prohlášení

Místopřísežně prohlašuji, že jsem celou bakalářskou práci vypracoval samostatně a všechny použité zdroje uvádím v seznamu použité literatury.

V Ostravě dne 10.05.2013 Podpis: …….

Břetislav Glac

OBSAH

1 Úvod

Pohlédnutím zpět do historie zjistíme, že počítače a počítačové programy se používaly pouze k velmi specifickým úkonům. Avšak vlivem stále rychlejšího technologického pokroku počítače postupně pronikly až do domácností. Celkově tento pokrok umožnil využití počítačů i v oblastech, kde to kdysi nebylo možné, ale dnes je to již nutností. Nenajdeme již žádný ekonomický sektor, který by informačních a komunikačních technologií nevyužíval, ba dokonce jejich použití nevyžadoval. Zpracování, přenos a uchovávání informací jsou dnes velmi důležité, jelikož správná a včasná informace přináší nejen výhody, ale i užitek. Prostředkem k tomu jsou informační systémy a jejich popisem, definicí a dělením se zabývá první část této bakalářské práce. Je zde také uvedeno, jakým způsobem jsou informační systémy dodávány a vyvíjeny a právě v oblasti jejich vývoje existují ve světě velmi rozličné až protichůdné názory.

Informační systémy lidem velice usnadňují práci. Pomocí nich jsou automatizovány různé procesy, což urychluje práci a snižuje míru lidských chyb. Toto je důvodem, proč informační systémy již nejsou výsadou velkých společností, ale jsou nasazovány i v malých firmách z různých oblastí. Jednou z těchto oblastí je i obchodní činnost, kde jsou informační systémy úspěšně využívány ve formě skladových, prodejních či pokladních systémů, a proto i menší obchody a prodejny začínají takové systémy využívat. Jedním z cílů této práce je zjistit jaká dostupná řešení v této oblasti jsou na českém trhu, tomu je věnována třetí kapitola, která zároveň posloužila jako inspirace pro čtvrtou kapitolu.

Čtvrtá kapitola popisuje návrh a implementaci právě prodejního a skladového systému. Je zde popsáno od samého začátku, jak takový systém vzniká a jaké věci se při tom musí řešit. V této kapitole jsou popsány jednotlivé dílčí cíle, které souvisí právě s vývojem systému. V prvé řadě je nutné analyzovat stávající systém, na jehož základě se poté začne budovat nový. Je nutné nahlédnout na to, jakým způsobem je momentálně dosahováno požadovaných funkcí, a zda je možné v novém systému některé prvky a aspekty starého systému zahrnout.

Druhý cíl představuje specifikace požadavků na systém. S pomocí zadavatele a jím pověřených osob je nutné definovat to, co bude muset systém nabízet, tedy jeho jednotlivé funkcionality a chování v určitých situacích. Zde je také nutné, alespoň slovně, nastínit i to, jak systém bude vypadat po grafické stránce.

Třetím cílem je samotný návrh systému. Na základě specifikovaných požadavků je třeba nahlížet na budoucí systém z mnoha úhlů pohledu. Musí se brát v úvahu fyzické rozmístění jednotlivých komponent systému nebo to jakým způsobem systém bude fungovat. Součástí cíle je i návrh datové struktury, nad kterou bude systém fungovat.

Čtvrtý cíl je již samotná implementace prodejního a skladového systému. K tomu je nutné vybrat správné nástroje, pomocí kterých bude dosaženo správné implementace. Pomocí těchto nástrojů a jednotlivých návrhů musí být vytvořeny jednotlivé komponenty systému. Implementace je ukončena testováním systému a jeho nasazením do ostrého provozu.

2 INFORMAČNÍ SYSTÉMY – VLASTNOSTI, TYPY, METODY VÝVOJE

Informační systém (IS) je natolik rozsáhlý pojem, že jej lze definovat více způsoby. Ty, které jej vystihují nejvíce, jsou uvedeny na následujících řádcích.

Informační systém je soubor lidí, metod a technologických prostředků, jejichž prostřednictvím je zabezpečován sběr, přenos, zpracování a uchování dat, tak aby bylo možno prezentovat informace uživatelům informačního systému [1].

Podle docenta Sedláčka [2, s. 5] „*Informační systém (Information system – IS) podniku je systém pro sběr, přenos, uchovávání, zpracování a poskytování dat (informací, znalostí) využívaných při činnosti podniku. Jeho komponentami jsou informační a komunikační technologie, data a lidé. Jeho cílem je efektivní podpora informačních, rozhodovacích a řídících procesů na všech úrovních řízení podniku. Vývoj a provoz jsou ovlivňovány organizačními, ekonomickými, právními, sociálními a dalšími aspekty.*“

Další z mnoha definic je taková, že informační systém je systémem mezi sebou propojených informací a procesů, jež pracují právě s těmito informacemi. V tomto ohledu je nutné proces chápat jako funkci zpracovávající vstupní informace na informace vystupující ze systému [3].

Dle výše zmíněných definic, a nejen těch, nemusí informační systém být automatizován počítačovými technologiemi, ale může být i pouze v papírové podobě. Informačním systémem může být například telefonní seznam, který uchovává informace o lidech, jejich adresách a telefonních číslech nebo účetní kniha, kde jsou zaznamenávány účetní operace a důležité údaje s nimi související.

V současné době se pro správný a účelný chod informačních systémů využívá informačních a komunikačních technologií (ICT), a proto takový systém označujeme jako IS/ICT. Informační a komunikační technologie jsou jak softwarové, tak hardwarové nástroje umožňující vyhledávat, modifikovat, manipulovat, přenášet či ukládat informace a zabezpečují komunikaci mezi lidmi a jednotlivými komponentami informačního systému [4].

Vzhledem k zaměření této práce je nutné zmínit i podnikový informační systém, jehož účelem je podpora podnikových procesů pomocí informačních a komunikačních technologií a

mnohdy je také prostředkem pro zvýšení celkové efektivity podniku. Prvky takového systému jsou lidé, data a ICT [5].

S informačními systémy jsou úzce spjaty i pojmy data a informace, které jsou mezi sebou mnohdy zaměňovány, či jejich význam bývá spojován, a proto jsou vysvětleny níže.

2.1 Data a informace

Data (singulár datum) jsou veškeré skutečnosti, jimiž informační systém disponuje a v informatice jsou označením pro text, čísla, zvuk nebo obraz. Při práci s daty, je nutné je rozdělit na strukturovaná a nestrukturovaná. Strukturovaná data jsou určitým logickým způsobem uspořádána, tudíž vyhledávání a vybírání těch důležitých z nich je snazší a efektivnější. Příkladem strukturovaných dat jsou relační databáze. Naproti tomu nestrukturovaná data nemají žádnou logickou strukturu nebo organizaci, příkladem jsou vizuální záznamy, audio nahrávky, textové dokumenty a podobně [6].

Samotná data však nedávají smysl, jsou jakoby vytržená z kontextu. Mít nějaký číselný údaj a nevědět co znamená, je bezpředmětné. U takových údajů je třeba zjistit jejich význam a účel a jejich zpracováním poté vznikají informace. Informace tedy poskytují určitou informační hodnotu, jsou to data užitečná, srozumitelná, použitelná a jejich důležitost a hodnota se projevuje v rozhodovacích procesech. Informace jsou daty, avšak to neplatí naopak [6, 7].

2.2 Vlastnosti informačních systémů

Aby informační systémy plnily své funkce, musí mít mnoho vlastností, které toto umožní. Základní vlastností informačního systému je, aby dokázal uspokojit informační potřebu uživatele. Ovšem nestačí, aby systém poskytl uživateli informace, uživateli musí být dodány ty správné informace a ve správném čase, pokud toto neplatí, informace ztrácí svoji hodnotu. Toto je fundamentální vlastnost, kterou musí informační systém disponovat. Pokud v rozhodovacím procesu bude použita informace, která je již neaktuální, neúplná nebo nepravdivá, pak toto může mít fatální důsledek na rozhodnutí. Jelikož se v posledních letech objem informací zvětšuje a konkurence mezi podniky je větší, tak je nutné, aby data byla navíc zpracovávána rychleji a efektivněji a informační výstupy byly přehledně a srozumitelně strukturovány [3, 6].

Obecně lze říci, že podnikový informační systém musí jeho uživatelům poskytnout funkce pro analytické, kontrolní, rozhodovací a plánovací činnosti a funkce pro evidenční a transakční operace. Dále přispívá ke zkracování, zjednodušování a snižování náročnosti podnikových procesů, tedy jejich racionalizaci. Také musí být bezpečný, spolehlivý a výkonný [8].

2.3 Typy informačních systémů

Informační systémy lze rozdělit do mnoha kategorií podle toho, jak je na ně nahlíženo. Mohou být rozděleny podle množství uživatelů systému, jeho velikosti, územního rozsahu, obsahu, účelu či zaměření. Na následujících řádcích budou uvedeny dvě, dle mého názoru nejdůležitější, možnosti jak klasifikovat informační systémy, a to podle organizačních úrovní podniku a podle vztahu k systému řízení.

2.3.1 Dělení podle organizačních úrovní podniku

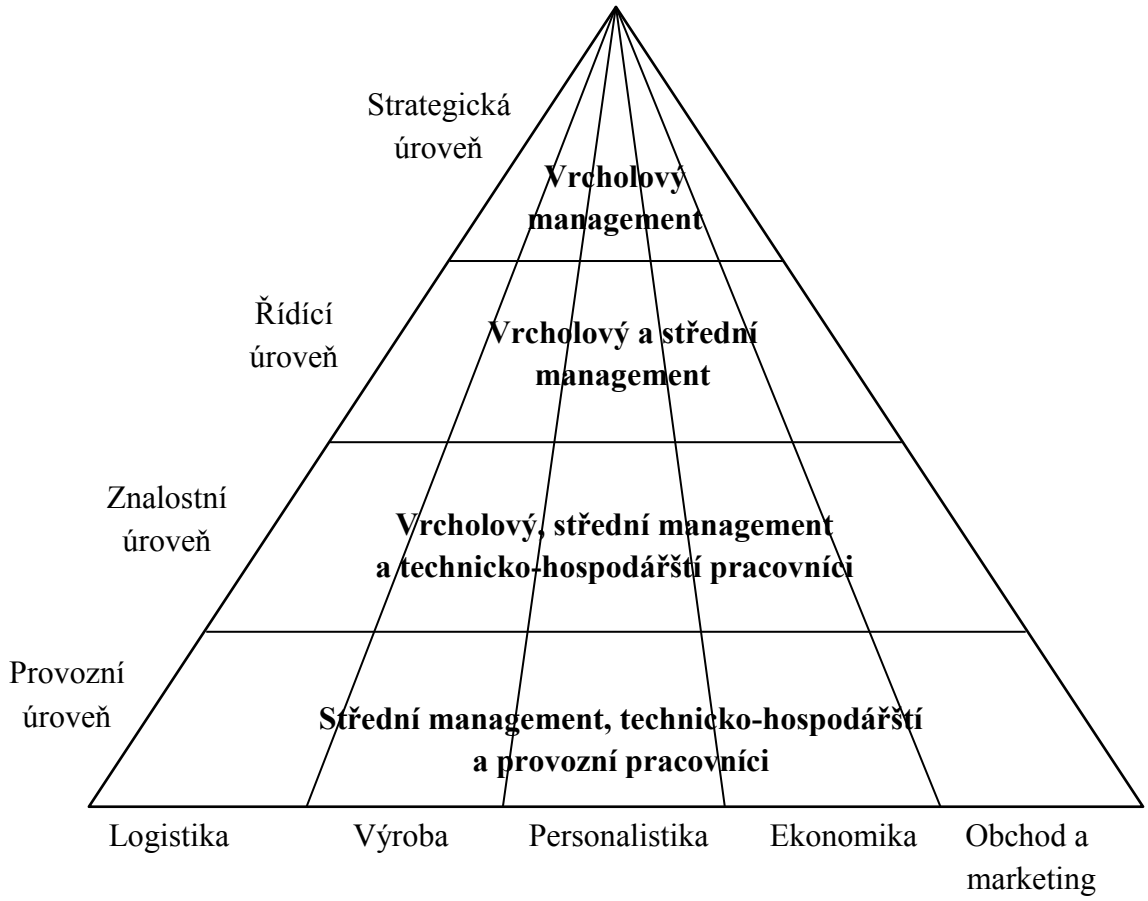
Každý podnik je členěn do několika organizačních úrovní a každá z těchto úrovní vyžaduje určitý druh informací a odlišný způsob jejich zpracování. Management potřebuje informace pro řízení podniku, ale žádná z úrovní podniku nedokáže poskytnout všechny informace a zároveň žádná z nich netvoří celistvý objekt, na jehož základě by byl nasazen informační systém. Obvykle se organizační úrovně dělí na provozní, znalostní, řídící a strategickou. Na obr. 2-1 je informační pyramida zobrazující tyto úrovně [7].

Provozní úroveň – informační systémy na této úrovni zpracovávají informace potřebné pro rutinní a každodenní činnosti podniku. Takové informace se mohou týkat prodeje a nákupu, zpracování zakázek, finančních transakcí apod. Je důležité, aby informace byly přesné, aktuální a dostupné [7].

Znalostní úroveň – na této úrovni je řízen tok dokumentů a rozšiřována znalostní základna podniku. Jsou zde zahrnuty kancelářské aplikace, aplikace pro podporu týmové práce (groupware) či klientské aplikace podnikového informačního systému. Informace této úrovně se týkají stavu hospodaření podniku, dodavatelů či zákazníků [7].

Řídící úroveň – zaměřuje se zejména na informace nutné pro administrativu a podporu rozhodování využívanými hlavně středním a vrcholovým managementem.

Informační systém této úrovně poskytuje strukturované či vizualizované informace, často pak v pravidelných intervalech. Příkladem mohou být finanční informace v určitém časovém úseku. Dalším typem výstupní informace jsou analytické nebo prognostické [7].



obr. 2-1: Informační pyramida podle organizačních úrovní podniku [7, s. 74].

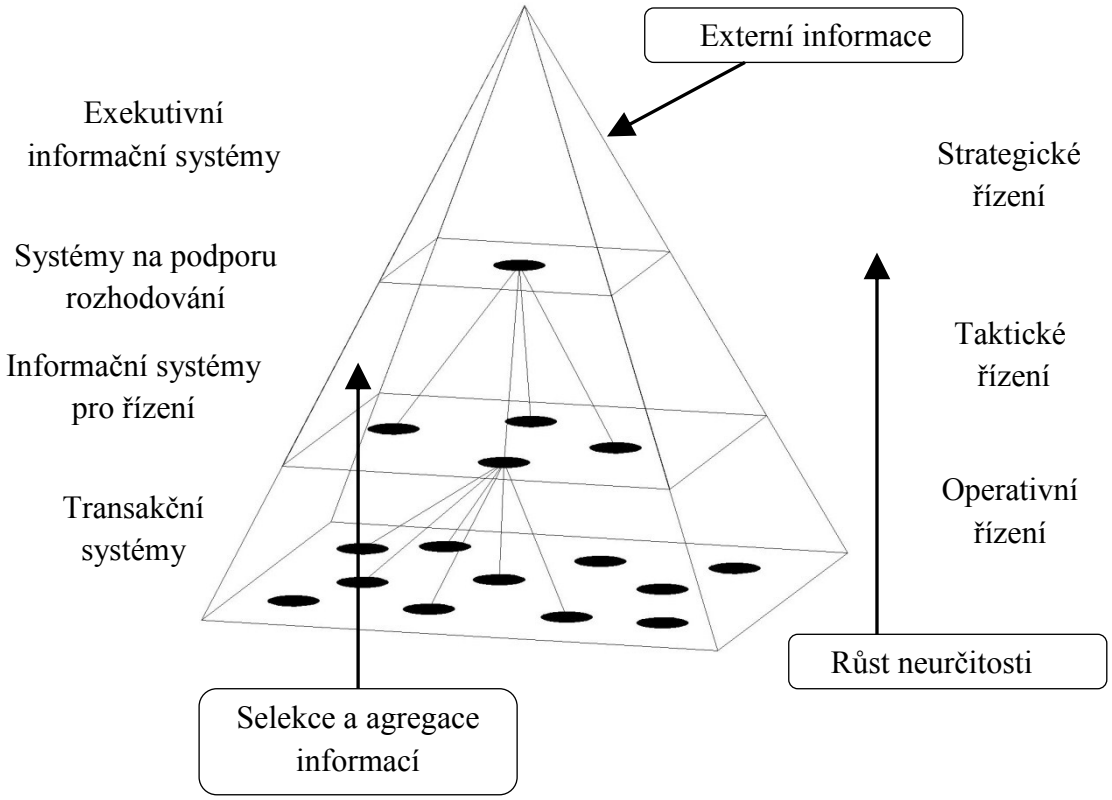
Strategická úroveň – informační systémy strategické úrovně jsou využívány vrcholovým managementem k odhalení očekávaných změn vně i uvnitř podniku a určení schopnosti podniku na ně reagovat. Také pomáhají zjistit vývoj dlouhodobých trendů [7].

2.3.2 Dělení podle vztahu IS k systému řízení

Jak bylo řečeno výše, informační systému se mohou dělit podle různých hledisek. Dělení podle vztahu IS k systému řízení mnoho z nich reflektuje. Na obr. 2-2 je vyobrazena informační pyramida, ve které rozlišujeme, na jaké úrovni se informační systém nachází. Toto dělení také můžeme považovat za globální architekturu IS/ICT podniku.

Transakční systémy (Transaction processing system – TPS) – se nacházejí na nejnižším stupni informační pyramidy, podporují nejnižší úroveň řízení. Jedná se o systémy,

které zpracovávají a automatizují typické rutinní úlohy na operativní úrovni, jako je účetnictví, skladové hospodářství, inventarizace či fakturace. Jelikož jsou jimi podporovány zejména úlohy probíhající v reálném čase, je zde kladen důraz na správnost a aktuálnost informací. Transakční systémy jsou zpravidla prvotním zdrojem dat podniku [1, 9].



obr. 2-2: Informační pyramida podle vztahu IS k systému řízení [1, s. 40].

Informační systémy pro řízení (Management infromation system – MIS) – jsou využívány středním managementem na taktické úrovni řízení podniku. Zdrojem dat jsou transakční systémy. Tato data jsou prezentována výstupy vytvářenými periodicky (týdně, měsíčně, ročně apod.) a mnohdy jsou určitým způsobem zpracovávána, např. agregována, sumarizována či filtrována. Takovým výstupům se říká reporty. Pomocí reportů si vedoucí pracovník může udělat představu o tom, v jakém stavu se podnik nachází, ale také v jakém stavu se nacházel v minulosti a podle těchto informací učinit nezbytná rozhodnutí [1, 9].

Systémy pro podporu rozhodování (Decision support system – DSS) – jsou úzce spjaty s informačními systémy pro řízení, jelikož zpracovávají stejná data, avšak požadavky na výstupy jsou neurčité a jsou odhalovány postupně v průběhu řešení úloh. Jsou využívány analytickými pracovníky a pracovníky managementu všech úrovní. Jednotlivé úlohy se neopakují, vždy dochází k určitým změnám v podmínkách řešení. Nad daty jsou prováděny

různé analýzy a mnohdy jsou výstupy prezentovány graficky, což má pro řídící pracovníky mnohem vyšší vypovídací schopnost než výstupy textové. Na základě takto získaných informací jsou pak prováděny simulace, podle kterých se pak podnik řídí. Výstupy jsou většinou vyžadovány ve velmi krátkém čase [1, 9].

Informační systémy pro vrcholové řízení (Executive infromation system - EIS) – slouží zejména vrcholovému managementu a jsou na nejvyšším stupni informační pyramidy (viz výše). Tyto systému zastávají integrační funkci mezi podřízenými systémy a jimi samými, jelikož využívají jejich informací a dat. Avšak vrcholový management nepotřebuje informace pouze o stavu vlastního podniku, ale také potřebuje informace z okolí podniku, a proto jsou tyto systémy navrhovány tak, aby umožňovaly přístup také k externím datům (data o konkurenci, o politické situaci, o bankovním sektoru apod.). Na rozdíl od předchozích systémů EIS podporují strategické rozhodování, tedy rozhodování v dlouhodobém měřítku, pomocí těchto systémů se vytvářejí analýzy nejen o současném stavu podniku, ale i o jeho budoucím stavu. Analýzy jsou pak většinou prezentovány graficky [1, 9].

2.4 Architektury informačních systémů

Architektura informačních systémů představuje jak grafický a verbální popis celého IS podniku. Pomocí architektury jsou popsány: struktura IS v souvislosti s organizační strukturou podniku, funkce IS související s jednotlivými podnikovými procesy, infrastruktury mezi jednotlivými částmi IS a vazby na okolí. Na architekturu však lze pohlížet různě. Pokud vezmeme v úvahu fyzické hledisko, pak architekturou je prostorové rozmístění hardwarových komponent, použité operační systémy či programové vybavení. Z logického hlediska se v systému určují vazby mezi komponentami či uzly systému a také to, jak jsou požadavky mezi nimi distribuovány. Je možné na architekturu pohlížet globálně, obecně jako na hierarchii podniku (viz kapitola 2.3.2, ve které jsou rozděleny informační systémy a toto dělení lze považovat i za obecnou globální architekturu sytému, na jednotlivých úrovních pyramidy se popisují dílčí části systému určitého podniku), nebo z dílčího pohledu, kdy se zkoumají jednotlivé části globální architektury [4, 10].

Architektura je prostředkem, který umožňuje komunikaci mezi vývojáři systému a vedením podniku, díky ní je možné si představit, jakým způsobem bude systém fungovat. Podle Voříška [4, s. 236] *„architektura vytváří relativně stabilní rámec řešení IS/ICT, do*

něhož se v průběhu vývoje postupně začleňují jednotlivé komponenty podle připraveného plánu a podle technologických, ekonomických a dalších možností – avšak s již předem definovanými vazbami na ostatní komponenty IS/ICT.“ Význam architektury se projevuje i poté co je IS vytvořen, pokud je architektura navržena účelně a správně, pak zohledňuje i budoucí potřeby integrovat nové komponenty namísto starých a již nevyhovujících, což je důležité při současném rychlém technologickém pokroku. Z ekonomického hlediska je prostředkem pro snižování nákladů, eliminuje množství chybných projektů a udržuje IS funkční po delší dobu, než kdyby byl vytvořen bez jasného konceptu. Zde je možné architekturu IS přirovnat ke stavebním plánům budovy, žádný stavitel nestaví budovu ani bez základního nákresu [4, 10].

2.4.1 Dílčí architektury

Dílčí architektury navazují na architekturu globální (viz výše) a blíže popisují fungování IS podniku. Rozlišujeme aplikační, datovou/informační, softwarovou a technologickou architekturu [4].

Aplikační architektura – popisuje jakým způsobem a jakým aplikačním softwarem jsou zabezpečeny jednotlivé funkce IS, jaké vazby jsou mezi jednotlivými aplikacemi a jaké vztahy jsou mezi aplikacemi a funkcemi, které podporují, zda aplikace podporuje jednu či více funkcí a zda funkce je podporována jednou nebo více aplikacemi [4].

Datová/informační architektura – je definována na datové základně IS podniku. Analýzou datových objektů a jejich vazeb se vytváří návrh datové základny. Výsledkem je schéma databází, které představuje tuto architekturu [11].

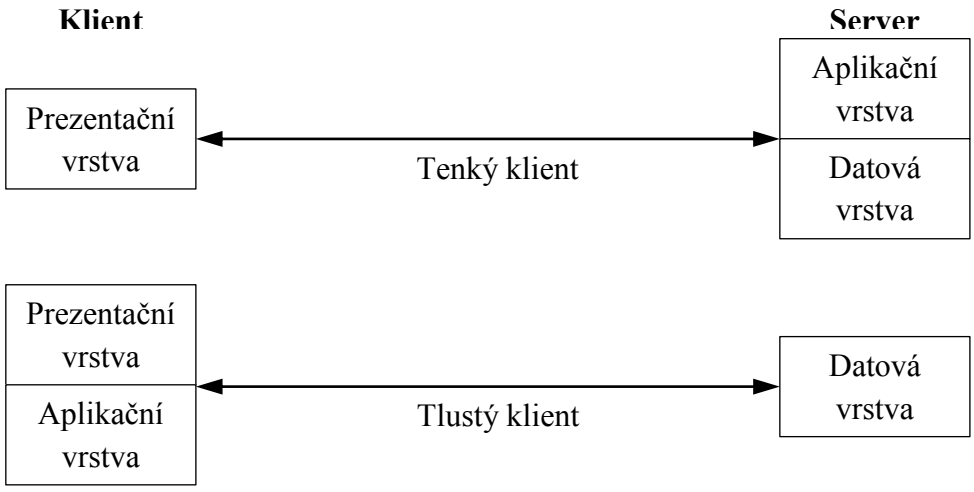
Technologická architektura – definuje vazby hardwarových, softwarových a datových komponent, způsob zpracování aplikací, způsob zpracování dat, popisuje vnitřní stavbu aplikací a jejich uživatelské rozhraní. Snahou architektů je v rámci podniku sjednotit technologickou architekturu jednotlivých aplikací systému, jelikož je klíčem ke snižování provozních nákladů [4, 11].

Softwarová architektura – se zaměřuje na jeden softwarový produkt (aplikace), u kterého popisuje, kolik programových modulů tvoří aplikaci, jejich uspořádání, specializaci, funkcionalitu a další aspekty. Uspořádání modulů větších aplikací může být lineární,

hierarchické, síťové nebo vrstvené. V rámci jedné aplikace rozlišujeme obecně tři funkce, datovou, věcně orientovanou a komunikační. Datová funkce zabezpečuje práci s daty (ukládání, výběr, zabezpečení), věcně orientovaná se stará o zpracování vstupních dat na výstupní a komunikační umožňuje komunikaci mezi uživatelem a aplikací podle toho, zda tyto funkce jsou poskytovány jedním či více samostatnými programy dělíme architekturu na monolitickou, dvouvrstvou, třívrstvou a případně i vícevrstvou. V monolitické architektuře jsou všechny tři funkce zabezpečeny jedním programem centralizovaně, v případě vícevrstvých aplikací je využito klient/server architektury (jedna aplikace figuruje jako klient a požaduje provedení nějaké funkce po druhé aplikaci, po serveru). Dále jsou popsány dvouvrstvá a třívrstvá architektura [4].

a) Dvouvrstvá architektura

Je typickým příkladem klient/server architektury. Jsou vytvořeny 3 vrstvy, aplikační, prezentační a datová, které mezi sebou komunikují právě jako klient se serverem. Prezentační vrstva přímo komunikuje s uživatelem, pomocí této vrstvy uživatel vidí výstupy, ovládá aplikaci apod. Aplikační vrstva je samotným jádrem aplikace, zabezpečuje věcně orientované funkce (viz výše). Datovou vrstvou jsou zabezpečeny datové funkce, jedná se o databázi, souborový systém apod. Dvouvrstvou architekturu můžeme rozdělit podle toho, zda je aplikační vrstva přítomna na serveru nebo u klienta, na tenkého klienta a tlustého klienta (viz obr. 2-3) [4].

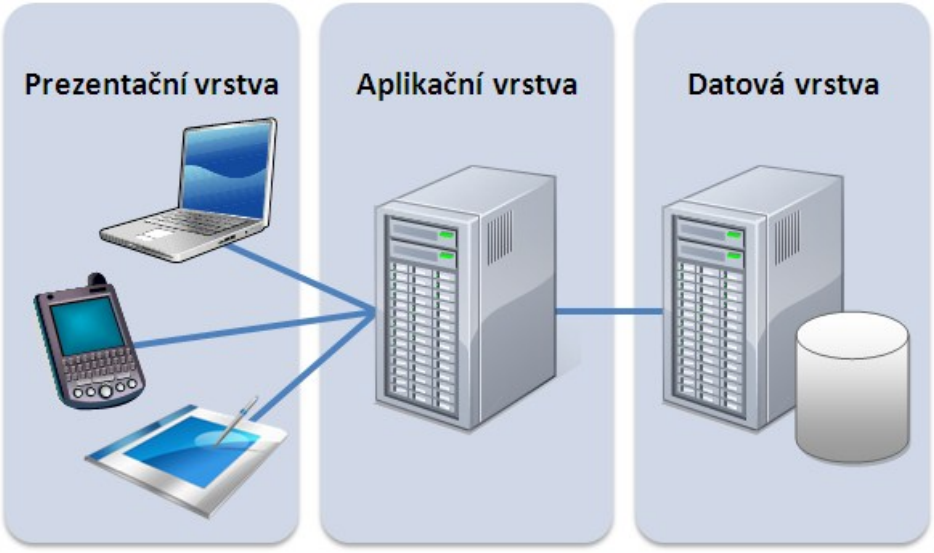


obr. 2-3: Varianty dvouvrstvé architektury [4, s. 274]

Tím, že se oddělí komunikační funkce od ostatních, je možné vytvořit aplikace pro různé platformy a operační systémy. Oproti monolitické architektuře je zde jednodušší údržba, jelikož jsou funkce odděleny, avšak ne úplně. Věcně orientované funkce jsou stále určitým způsobem propojeny s nějakou další skupinou funkcí [4].

b) Třívrstvá architektura

Všechny tři skupiny funkcí (datové, věcně orientované a komunikační) jsou stejně jako u dvouvrstvé architektury rozděleny do tří vrstev, avšak oproti dvouvrstvé architektuře jsou všechny na sobě nezávislé. Na straně klienta je zde pouze prezentační vrstva, zbývající dvě jsou na straně serveru. Třívrstvá architektura zahrnuje veškeré výhody dvouvrstvé architektury, navíc údržba je ještě jednodušší, jelikož vrstvy jsou odděleny od sebe kompletně. Z ekonomického hlediska je tato architektura náročnější na vstupní náklady, je nutné pořídit servery a klientské stanice, avšak do budoucna je prostředkem ke snižování nákladů, jelikož prezentační vrstva je poměrně nenáročná na hardware a může běžet i na levnějších zařízeních. Architektura je také velice flexibilní vůči technologickému pokroku, software i hardware je možné vybrat pro každou vrstvu zvlášť a uživatelským požadavkům, kdy je například možné změnit uživatelské rozhraní aplikace bez nutnosti zasahovat do ostatních částí aplikace. Typickým příkladem třívrstvé architektury je využití tenkého klienta pro prezentační vrstvu, aplikačního serveru pro aplikační vrstvu a databázového serveru pro datovou vrstvu [4].



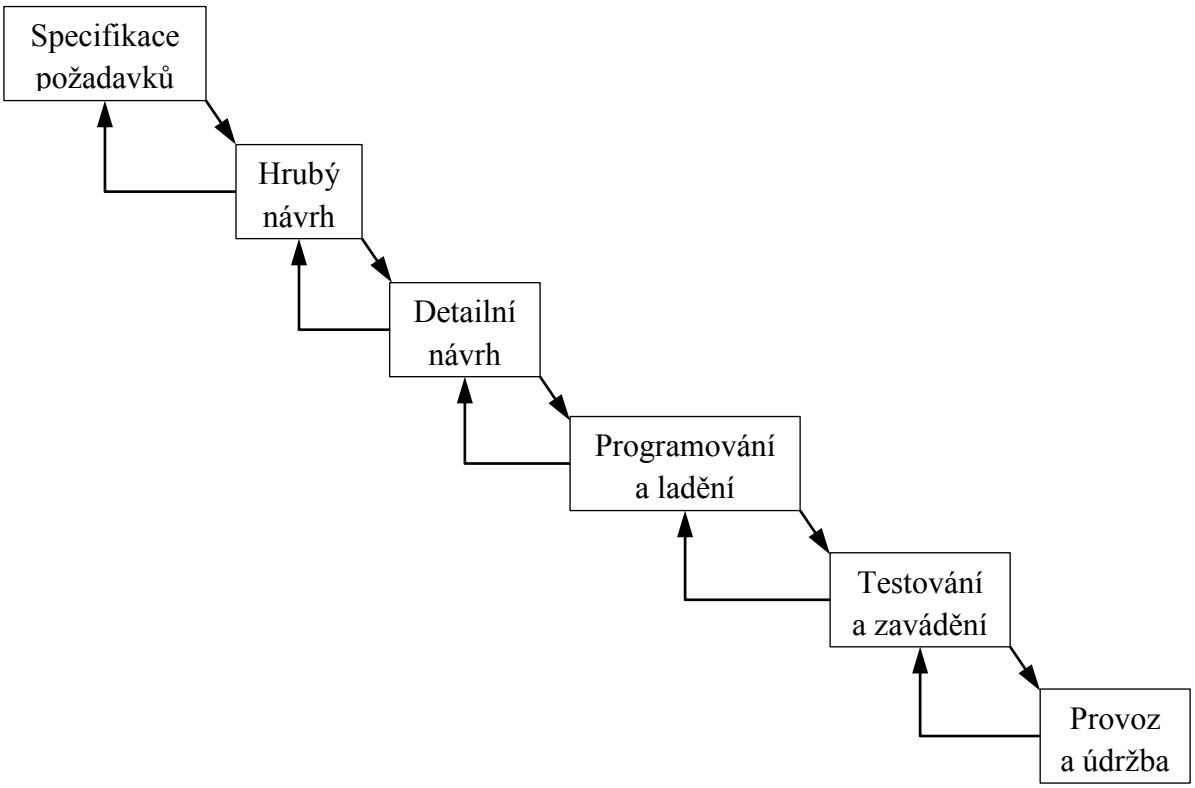
obr. 2-4: Třívrstvá architektura [29]

2.5 Životní cyklus informačních systémů

Každý informační systém musí vzniknout, ale také zaniknout. Časový úsek mezi rozhodnutím vytvořit informační systém až po jeho zánik se nazývá životní cyklus informačního systému. Tento časový úsek lze rozdělit na několik dílčích procesů, etap či fází, kterými informační systém prochází. U různých autorů se můžeme setkat s různým počtem etap, přičemž tento počet se pohybuje mezi čtyřmi až osmi fázemi, a také různým pojmenováním jednotlivých etap. Jednou z variant členění životního cyklu je plánování (specification, inception), návrh (design), zavádění (implementation) a provoz s údržbou (operation and maintenance). Dodnes se informační technologie mění a v souvislosti s tímto vznikly či určitým způsobem byly měněny modely životního cyklu. Tyto modely jsou úzce spjaty s metodikami vývoje IS (viz níže), jelikož různé metodické přístupy nahlížejí na IS jiným způsobem. Dále jsou popsány některé modely: vodopádový, prototypový, spirálový a inkrementální [1, 4, 9].

2.5.1 Vodopádový model

Patří mezi modely nejstarší, je prvním modelem, který se hojně rozšířil, a to v 70. a 80. letech 20. století. Tento model představoval významný pokrok, jelikož rozdělil vývoj do několika samostatných fází, a tím umožnil celý proces vývoje uspořádat a znovu opakovat. Jednotlivé po sobě jdoucí fáze jsou specifikace požadavků, analytická fáze, kdy se nejprve systém navrhuje obecně a poté detailně, implementační fáze, kdy se programují a ladí jednotlivé komponenty systému, následuje testování a zavádění a poslední fází je samotný provoz a údržba. Hlavními cíli modelu jsou zvýšení disciplíny (zavedení standardů a jednotné dokumentace, aby se tvořili přehledné zdrojové kódy a vývojové diagramy), možnost řešení složitějších problémů, zvýšení spolehlivosti a snadnost opravení chyb (na konci každého stupně se provádějí kontroly, aby se případné chyby zachytily co nejdříve) a zvýšení využití zdrojů (lidských i finančních díky prováděným kontrolám nákladů v jednotlivých fázích). Svůj název získal podle grafického zobrazení (viz obr. 2-5), kde jednotlivé fáze na sebe přímo navazují a připomínají vodopád. Pořadí jednotlivých fází je pevně stanoveno a jejich výsledky slouží jako podklady pro etapy přímo následující. Je nutné dodržet to, že do další fáze je možné se přesunout až je předcházející fáze ukončena a pokud se vyskytnou nějaké nedostatky, je možné vrátit se pouze k předcházející fázi [1, 4].



obr. 2-5: Vodopádový model [1, s. 60]

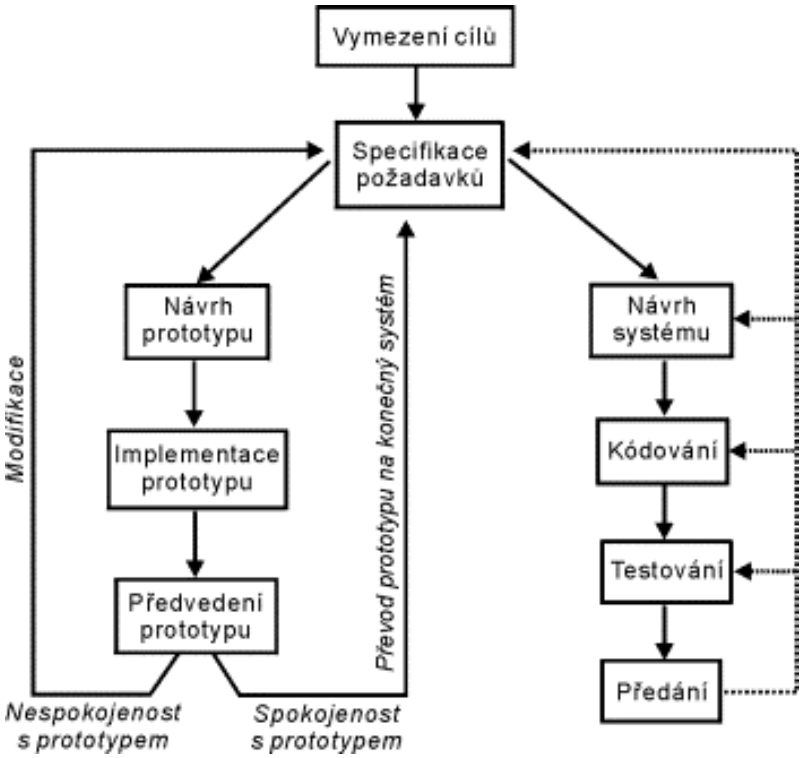
Tento model je výhodné nasadit v projektech, kdy jsou známy veškeré požadavky na systém již v začátcích jeho vývoje. Nehodí se tedy na projekty, kdy je v zadání projektu velká míra abstrakce a možnost, že se zadání může určitým způsobem změnit. Velkou nevýhodou je integrace jednotlivých modulů, která se provádí až po jejich naprogramování. To s sebou nese velká rizika zjištění nedostatků vyžadujících změnu návrhu systému a tím i celkovému zpoždění celého projektu. Navíc objednatel systému je do projektu zapojen hlavně na jeho začátku a konci, přičemž v průběhu nemá velkou možnost dozoru nad projektem [4].

Podle Doucka [9] je v tomto případě většina nákladů spojených s vývojem a provozem systému na straně objednatele systému. Objednatel hradí veškeré náklady na projektu od jeho přípravy až po provoz a údržbu. Na dodavateli je pouze pokrytí nákladů spojených s vyškolením vlastních zaměstnanců a se zajištěním potřebných informačních technologií [9].

2.5.2 Prototypový model

Model vychází z předpokladu, že během procesu vývoje IS se budou měnit výchozí požadavky objednatele a je formulován tak, aby na tyto změny bylo možné reagovat, čímž se podstatně liší od vodopádového modelu. Model se začal objevovat v 80. letech minulého století, jelikož objednatelé se dožadovali implementace alespoň části IS v co nejkratším čase.

Cílem modelu je tedy vytvořit určité jádro systému, tzv. 1. prototyp, a to poté vylepšovat a rozšiřovat podle potřeb objednatele. Tento prototyp je navržen a implementován co nejdříve a s takovou funkcionalitou, aby se objednateli mohlo prezentovat něco, na co může zareagovat a k čemu může mít případné připomínky. Jak probíhá vývoj a jaké jsou jednotlivé fáze je zachyceno na obr. 2-6 [1, 3].



obr. 2-6: Prototypový model [3]

Model odstraňuje základní nedostatek vodopádového modelu, kdy se objednatel zapojoval do vývoje na samotném počátku a poté až na jeho konci. Díky tomu je možné přesně popsat požadavky objednatele, případně reagovat na jejich změny. Nevýhodou je, že u rozsáhlejších projektů je jejich vedení náročné. Důvodem je, že zde nejsou přesně stanoveny hranice, kdy začínají a kdy končí jednotlivé fáze, ty se mezi sebou překrývají. Důsledkem toho pak dodavatel systému musí pečlivě zvážit, zda změnám požadavků ze strany objednatele lze či nelze vyhovět [1].

2.5.3 Spirálový model

Spirálový model byl poprvé definován Barrym Boehmem v roce 1985. Jedná se o model, který odstraňuje největší nedostatky vodopádového modelu, jako jsou nízká míra zapojení objednatele systému a nutnost definovat celý projekt co nejpřesněji již na samém počátku životního cyklu. Model zavádí analýzu rizik, na rozdíl od předchozího modelu, a

iterativní přístup k vývoji systému, tím jej umožňuje rozdělit do menších celků a díky tomu na počátku vývoje nemusí být známy úplné a detailní požadavky na systém, ty se upřesňují po jednotlivých iteracích. V rámci jedné iterace probíhají podobné fáze jako u vodopádového modelu, což je patrné z obr. 2-7, tedy definice cílů, analýza rizik, návrh řešení, ověření, vývoj, testování a plánování. Z obrázku je také patrné, že jsou uskutečněny čtyři iterace, avšak jednotlivé iterace se opakují do té doby, než je výsledný systém hotov. V první iteraci jde zejména o to zjistit globální rizika celého projektu a specifikovat základní požadavky na systém. V dalších iteracích jsou pak rizika i požadavky nadále upřesňovány [12].

obr. 2-7: Spirálový model [9, s. 72]

Tím že byl v modelu využit iterativní přístup, tak se testování provádí již v počátcích samotného vývoje, což umožňuje odchytit chyby systému velmi brzy a také to umožňuje pozdější detailnější specifikaci cílů v průběhu projektu, a proto se tento model hodí pro větší projekty než model vodopádový. Navíc každá iterace s sebou přináší již nějaké funkční řešení, které je možné konzultovat s objednatelem systému, jenž je tedy ve vývoji zainteresován více než u předchozího modelu a sám může blíže specifikovat své požadavky na základě již vytvořené části systému. Tedy zpětná vazba mezi objednatelem a dodavatelem systému neprobíhá pouze na konci celého vývoje, ale v jeho průběhu a tím se snižuje míra rizika s nespokojeností objednatele. To ovšem může být nevýhodou, pokud se objednatel vývoje

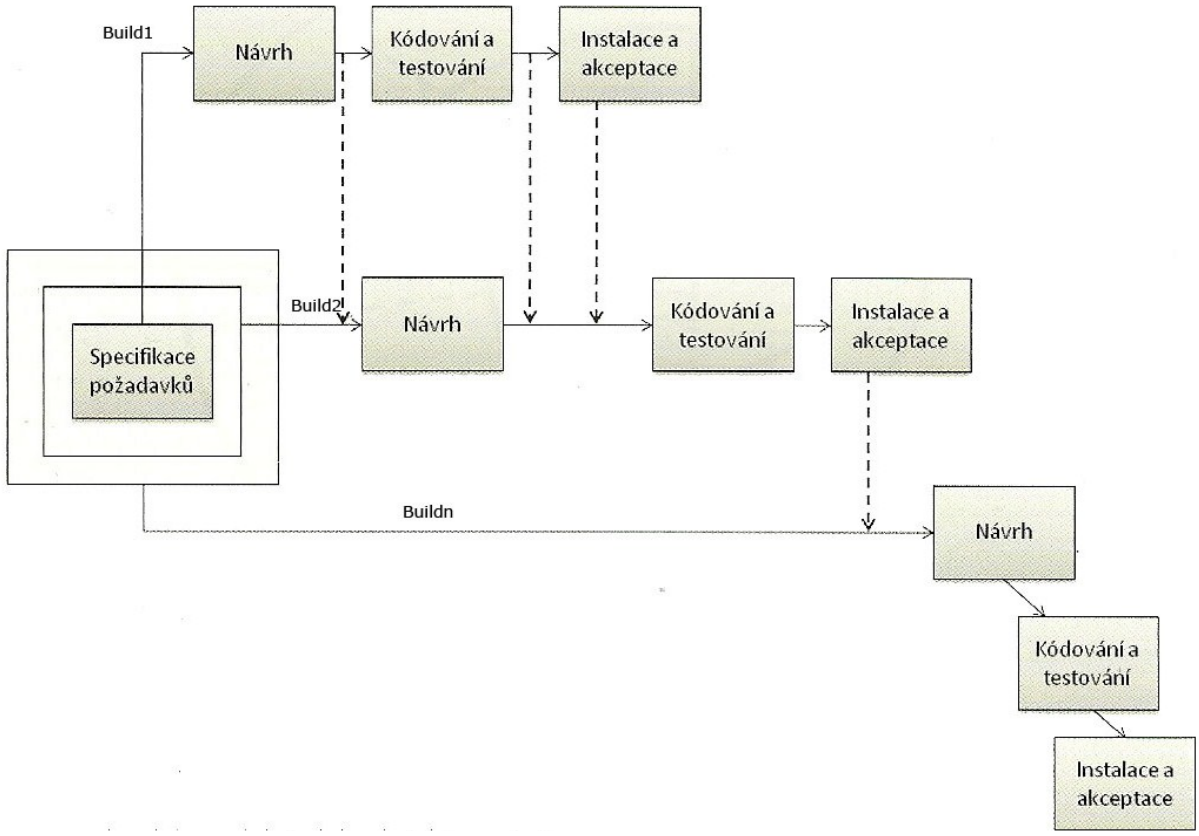
neúčastní. Další velkou výhodou je, že integrace jednotlivých programových modulů probíhá koncem každé integrace a jak již bylo řečeno, tak případné chyby a nedostatky jsou odhaleny včas [1, 4, 13].

Z ekonomického hlediska jsou náklady rozděleny mezi objednatele a dodavatele systému přibližně na polovinu, často pak připadá o něco větší část nákladů na dodavatele. Je to dáno tím, že když dodavatel přijde na trh s takto vytvořeným prototypovým řešením, které vytvořil pro určitou organizaci, tak jej nabízí dalším zájemcům, kde toto řešení implementuje a celé řešení nadále vyvíjí a přináší nové verze. Skutečnost je taková, že dodavatel systému sice má na své straně většinové náklady, avšak předpokládá, že systém bude prodán vícekrát a tedy náklady na systém tímto vyváží. S každou novou verzí, kterou dodavatel vyvíjí pro nové objednatele, pak přichází nové části systému, které buď na základě specifikací musí dodavatel vytvořit a poté je bude v nových verzích dodávat, anebo jsou na přání objednatele s tím, že je nemůže opakovaně použít [9].

2.5.4 Inkrementální model

Principem inkrementálního modelu je na začátku hrubě definovat celý systém a poté jej rozdělit na několik dílčích částí, tzv. přírůstků, jež jsou realizovány samostatně, přičemž každý z přírůstků prochází jednotlivými fázemi vývoje nezávisle na ostatních (viz vodopádový a spirálový model), což je patrné z obr. 2-8. Přírůstek v tomto smyslu představuje jednu určitou část systému. Přírůstky na sobě mají již předem definované vazby, ale pro každý z nich může být využit jiný model životního cyklu, či jiná metodika vývoje (viz kapitola 2.7). V předchozích modelech jednotlivé fáze na sebe přímo navazovaly, avšak v tomto modelu nejenže na sebe navazují, ale také se překrývají během vývoje jednotlivých přírůstků. Není přímo dané, že jednotlivé přírůstky musí být vyvíjeny až poté, co předchozí jsou dokončeny. Jsou tedy vyvíjeny buď poté, co se zjistí, že jsou nutné, nebo paralelně spolu s ostatními, když je známo, že jsou potřeba [4, 12].

Výhoda modelu tkví v tom, že objednatel v průběhu vývoje může dohlížet na to, zda se řešení vyvíjí podle jeho představ a dodavatel může získat zpětnou vazbu velice rychle. Dále umožňuje přizvat dalšího či změnit dosavadního dodavatele systému, ale to pouze v rámci různých přírůstků, jelikož lze vyvíjet určitý přírůstek nezávisle na ostatních. Na druhou stranu ne vždy je možné rozdělit celý systém na jednotlivé přírůstky [4, 12].



obr. 2-8: Inkrementální model [12, s. 52]

Z ekonomického hlediska je tento model velmi výhodný, jelikož je možné vývoj celého informačního systému přerušit a odložit v případě, že se vyskytnou finanční problémy. Takový systém sice není plně funkční, ale jednotlivé přírůstky je možné již využívat [9].

2.6 Varianty vývoje informačních systémů

„Vývoj IS je proces, jehož cílem je dosažení plánované změny informačního systému podniku. Změna se může týkat kterékoliv komponenty IS (nová aplikace, změna technologické infrastruktury apod.). Podstatné změny se realizují projektem. Ukončením projektu vzniká nová verze IS podniku. Z pohledu byznysu jsou nejpodstatnějšími změnami ty, při kterých vzniká nebo se podstatně mění softwarová aplikace ovlivňující průběh byznys procesu, nebo data v něm zpracovávaná.“ [4, s. 80]

Podnik musí řešit několik základních otázek při vývoji a nasazení nového informačního systému. První otázkou je, jakým způsobem budou jednotlivé aplikace informačního systému vyvinuty či získány. Dále zda aplikace budou vyvíjeny či zakoupeny samostatně, anebo budou součástí softwarového balíku. Poslední otázkou, kterou podnik musí vyřešit je, zda vývoj a provoz IS bude probíhat ve vlastní režii nebo pomocí cizích zdrojů [4].

Existují dva základní způsoby, jak získat, respektive vyvinout novou aplikaci, a to individuální aplikační software (IASW) nebo typový aplikační software (TASW).

IASW – znamená, že aplikace jsou vyvíjeny podle potřeb podniku a jednotlivé funkce jsou navrhovány tak, aby plně podporovaly činnosti podnikových procesů, pro které jsou navrženy. IASW přináší podniku velkou konkurenční výhodu, jelikož aplikace jsou jedinečné a optimalizované přesně podle potřeb podniku, avšak často náklady bývají mnohdy vyšší než při použití TASW a trvá delší dobu takový software vyvinout [4].

TASW – jsou již navržené a vytvořené aplikace specializovaným výrobcem a to tak, že výrobce implementuje do aplikace již ověřené a osvědčené postupy a funkce, které jsou v daném oboru známy. Výhodou takového nasazení je to, že pořízení licence takových aplikací je levnější než vlastní vývoj IASW, a to i přes fakt, že vývoj takových aplikací je mnohem nákladnější, ale náklady na vývoj se rozdělí mezi více uživatelů. Na druhou stranu aplikace obsahují i funkce, které podnik nepotřebuje a platí tedy za to, co nevyužije, případně funkce není navržena tak, aby plně podporovala podnikový proces, který se pak aplikaci musí přizpůsobit. Proto se využívá činností, které se snaží nevýhody eliminovat, těmi jsou lokalizace, customizace, integrace a personalizace. Lokalizací se rozumí úprava softwaru podle země, kde je software použit, tedy podle její legislativy, jazyka či kulturních zvyklostí. Customizace je úprava aplikace podle potřeb zákazníka a podle jeho požadavků, je upravován vzhled grafického rozhraní a výstupních sestav, funkce jsou přizpůsobeny podnikovým procesům apod. Integrace je propojení nové aplikace s již stávajícími komponentami IS podniku. Personalizace pak je úprava aplikace podle jednotlivých uživatelů. Uživatelé si tak mohou měnit grafické rozhraní, jazyk aplikace apod. [4]

Samostatný vývoj jednotlivých aplikací (komponent) pro IS podniku znamená, že každá funkce IS bude pokryta nejlepší dostupnou aplikací, avšak integrace jednotlivých komponent musí být důkladně ošetřena. Dále pak podnik není závislý na jediném dodavateli řešení, ale žádný z dodavatelů pak nenese odpovědnost za integritu celého systému. Softwarový balík pak má přesně opačné výhody a nevýhody [4].

Poslední otázkou je zda podnik bude vyvíjet a provozovat systém vlastními silami nebo pomocí cizích zdrojů. Zde se nejvíce nahlíží na náklady spojené s pořízením a provozem, spolehlivost systému a bezpečnost dat. Vlastní vývoj bývá nákladnější a časově

náročnější, a proto v dnešní době podniky toto řeší cizími zdroji, avšak samotný provoz si již zajišťují samy [4].

Zodpovězením výše uvedených otázek a zohledněním jednotlivých výhod a nevýhod jednotlivých možností pak podnik vybere tu variantu, která je pro něj nejvýhodnější a nejpřínosnější.

2.7 Metodiky vývoje informačních systémů

„Metodika tvorby IS je doporučený souhrn etap, přístupů, zásad, postupů, pravidel, dokumentů, řízení, metod, technik a nástrojů pro tvůrce informačních systémů, který pokrývá celý životní cyklus informačních systémů. Určuje kdo, kdy, co a proč má dělat během vývoje a provozu IS.“ [14, s. 7]

V současnosti existuje velké množství metodik budování IS/ICT a každá z nich pak je vhodná pro jiný typ projektu. Buchalcevová [13] popsala kritéria, jež lze použít pro kategorizaci metodik: zaměření metodiky, rozsah metodiky, váha metodiky, typ řešení, doména a přístup k řešení.

Zaměření metodiky – toto kritérium rozlišuje, zda je metodika zaměřena na budování IS/ICT v rámci celé organizace, anebo pouze na jednotlivý projekt, tedy jestli se jedná o globální nebo projektovou metodiku [4, 13].

Rozsah metodiky – je obyčejně chápán jako počet fází životního cyklu informačního systému pokrývaného metodikou. Pro potřeby rozdělení metodik jsou zohledněny tři oblasti, jejichž průnik udává rozsah metodiky, těmi jsou fáze životního cyklu, role a dimenze. Oblast fáze životního cyklu určuje, kterými fázemi se metodika zabývá. Rolí se rozumí, jací specialisté (např. uživatel, programátor, tester, správce databáze, vedoucí projektu ad.) a specializované skupiny (doménová role, role zájmových skupin, softwarové inženýrské role v projektu ad.) musí být v projektu zahrnuty. Metodiky se pak liší počtem a typem různých rolí. Dimenze je možnost nahlížet na vývoj IS z různých pohledů, rozlišujeme organizační, datovou, technologickou a procesní dimenzi [13].

Váha metodiky – velikost, hustota a váha metodiky jsou pojmy odvozené od charakteristik metodiky označované zkratkou PARTS (Precision – podrobnonost, Accuracy –

přesnost, Relevance – relevance, Tolerance – tolerance a Scale – měřítko). Velikost označuje počet kontrolních prvků v metodice a hustota míru podrobnosti a těsnost tolerance metodiky, požadovanou detailnost a konzistenci prvků. Váha metodiky je součinem těchto dvou veličin. Rozlišujeme těžké (rigorózní) a lehké (agilní) metodiky [13].

Typ řešení – při vytváření nového IS není vždy nutné vytvářet nový software. Toto kritérium zohledňuje více možností, jak získat software, a to vývojem nového řešení, integrací již vytvořeného řešení, rozvojem a rozšířením (upgradem) stávajícího řešení, customizací a implementací typového řešení (TASW) a užitím řešení (outsourcing) [13].

Doména – toto kritérium lze využít pouze u projektových metodik. Doménou se rozumí určitá oblast nebo skupina, pro kterou je informační systém vytvářen. Jsou rozlišovány například tyto domény: Customer Relationship Management (řízení vztahu se zákazníky, Content Management (řízení obsahu), Enterprise Application Integration (integrace podnikových aplikací, Supply Chain Management (řízení dodavatelských řetězců) ad. [13]

Přístup k řešení – jak tvrdí Buchalcevová [13, s. 27] „*toto kritérium zohledňuje základní paradigma, na kterém je metodika založena.“* Jsou rozlišovány metodiky pro strukturovaný vývoj, rychlý vývoj aplikací (RAD), objektový vývoj, komponentový vývoj nebo vývoj orientovaný na služby [4].

V dnešní době je možné setkat se se dvěma hlavními metodickými přístupy k vývoji informačních systémů. Těmi jsou rigorózní (tradiční, těžké) a agilní (lehké) metodiky. Hlavním, ale ne jediným, kritériem, ve kterém se tyto proudy liší, je váha metodiky [13].

2.7.1 Rigorózní metodiky

Tradiční přístupy nahlížejí na tvorbu IS jako na jasně definovaný proces, který lze měřit, řídit plánovat, důkladně popsat a podle popisu jej pak opakovaně realizovat a vylepšovat. Z toho důvodu jsou zde veškeré činnosti a procesy přesně a jasně definovány. Je nutné podotknout, že většina rigorózních metodik vychází z vodopádového modelu (viz výše) životního cyklu informačního systému, i když je možné setkat se s tradičními metodikami vycházejícími z iterativního nebo inkrementálního modelu. Tyto metodiky jsou vhodné zejména pro rozsáhlejší projekty s předem definovanými cíli [4, 13].

Mezi rigorózní metodiky patří např. Rational Unified Process (RUP) nebo Microsoft solutions Framework (MSF). RUP je metodika založena na iterativním vývoji (zejména na spirálovém modelu životního cyklu IS) a zaměřuje se na architekturu softwaru, řízení požadavků a návrh systému. Také již předem podrobně popisuje prvky použité při vývoji systému, např. procesy, činnosti, produkty ad. Popisuje čtyři fáze, které probíhají v jednotlivých iteracích, počáteční fáze (inception), elaborační fáze (elaboration), konstrukční fáze (construction) a fáze nasazení (transition) [4, 13].

Naproti tomu MSF je spíše metodický rámec, který se stále vyvíjí a na jehož základě lze definovat metodiku pro každou organizaci či projekt zvlášť. MSF rozlišuje dva procesy – týmový a procesní. Týmový model se zaměřuje na organizační strukturu projektového týmu, rozlišuje sedm skupin pracovníků, na jejichž základě jsou poté definovány role: zkušenosti uživatelů (analytik), řízení produktu (produktový manager), řízení projektu (projektový manager), architektura (architekt), vývoj (vývojář), testování (tester) a release/provoz (release manager). Procesní model je definován jako iterativní, přičemž iterace jsou krátké a jejich výsledkem je produkt s vylepšenou nebo přidanou funkčností. Každá iterace obsahuje pět základních fází: tvorba vize, plánování, vývoj, stabilizace a nasazení. Fáze jsou ukončeny kontrolním bodem, tzv. checkpoint, kdy se hodnotí dosavadní průběh projektu a rozhoduje se o jeho budoucnosti [4, 12].

Obě výše popsané metodiky jsou v základu rigorózní metodiky, avšak v rámci jejich vývoje byly do nich přidávány také agilní praktiky. V dnešní době se tyto metodiky řadí na pomezí agilního a rigorózního přístupu [12].

2.7.2 Agilní metodiky

Agilní přístupy se nesnaží proces vývoje IS popisovat, ale považují tvorbu IS za proces založený na zkušenostech, který je třeba neustále sledovat a upravovat podle skutečnosti. Tudíž jednotlivé metodiky popisují principy a praktiky použité při vývoje softwaru a ne jednotlivé procesy. Agilní metodiky se začaly prosazovat v druhé polovině 90. let minulého stol. z důvodu, že se začalo požadovat stále rychlejší a dřívější zavedení IS. Tomuto požadavku rigorózní metodiky však nevyhovují. Hlavním principem je vytvořit alespoň část systému v co nejkratším čase, na něm prověřit funkčnost, předvést objednateli a podle zpětné vazby jej pak upravit [12].

V únoru 2001 se v Utahu sešlo sedmnáct představitelů těchto metodik, kde společně vytvořili a podepsali „Manifest agilního vývoje software“ a byla vytvořena Aliance agilního vývoje (Agile Alliance). V manifestu stojí: „*Odhalujeme lepší cesty jak vyvíjet software tím, že jej tvoříme a pomáháme s jeho tvorbou ostatním. Při této práci jsme dospěli k těmto hodnotám: individualita a interakce před procesy a nástroji, fungující software před vyčerpávající dokumentací, spolupráce se zákazníkem před jednáním o smlouvě, reakce na změny před dodržováním plánu. I když body na pravé straně mají hodnotu, ceníme si bodů na levé straně více.“* [15]

Výše uvedenými hodnotami, nebo alespoň jejich pozměněnými verzemi, se řídí všechny agilní metodiky. Mezi ně patří např. Feature driven development (FDD), metodika Scrum nebo Extrémní programování (XP) [12].

Podle průzkumu z roku 2012 společnosti VersinOne je nejrozšířenější agilní metodikou Scrum, kterou používá 54% respondentů, na druhé příčce s 12% pak je kombinace metodiky Scrum a Extrémního programování. Samotné Extrémní programování a Feature Driven Development využívají 2% respondentů. Z celkového počtu 4048 respondentů jich 84% uvedlo, že jejich společnost využívá agilního vývoje, což je nárůst o 4% oproti předchozímu roku. Agilní přístupy jsou tedy rozšířenější a více využívány oproti tradičním přístupům [16].

3 ANALÝZA NABÍDKY KONKURENČNÍCH SYSTÉMŮ A APLIKACÍ

Tato práce je primárně zaměřena na prodejní (pokladní) a skladové systémy, a proto tato kapitola pojednává o těchto systémech a jejich dostupnosti na trhu. Na trhu je velká nabídka takových systémů, proto níže budou některé z nich uvedeny, budou popsány jejich funkce, vlastnosti a další specifika.

3.1 Pokladní software Conto

Tento systém obsahuje kromě pokladního také skladový systém, přičemž oba spolupracují mezi sebou. Conto je vhodný jak pro různá restaurační zařízení, tak i pro obchody s přímým prodejem, je tedy variabilní.

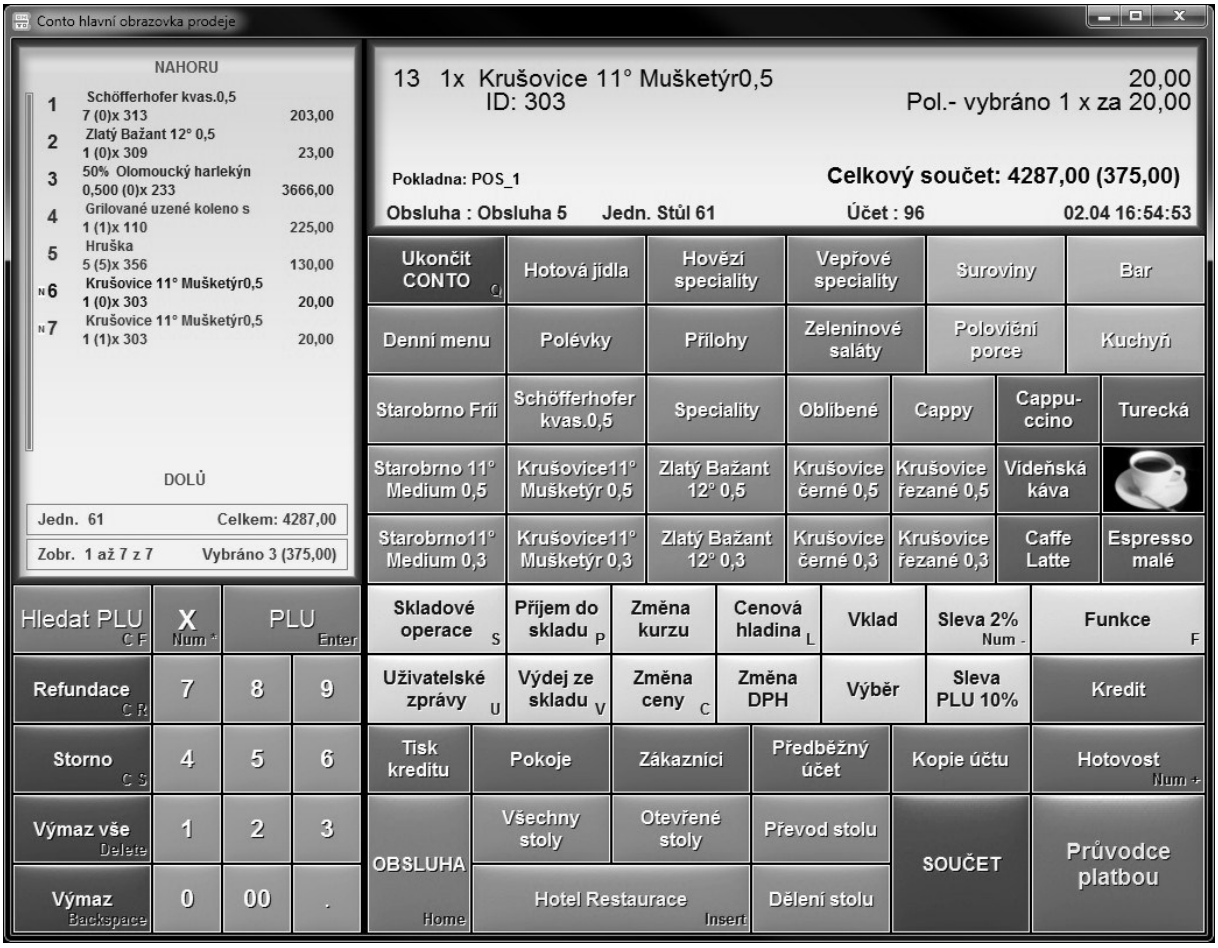
Grafické uživatelské rozhraní je možné nastavit podle potřeb, přičemž je přizpůsobeno pro použití s dotykovým monitorem i s klasickou klávesnicí. Systém umožňuje použití periferních zařízení, jako jsou scannery čárových kódů, registrační pokladny nebo tiskárny, čímž se usnadňuje použití systému. Je umožněn tisk účtenek různých formátů podle použité tiskárny.

K systému je možné dokoupit modul statistik prodejů, který umožňuje analýzy prodejů v různých časových obdobích, analýzy skladových zásob a jejich pohyby, různé možnosti filtrování ad.

Systém je možné použít lokálně na jednom počítači anebo síťově, kdy je možné k serveru, kde je nainstalován aplikační a databázový software, připojit klientské stanice. Díky tomu je možné používat najednou více pokladen. Je zde obsažena správa uživatelských účtů, každý účet se řadí do skupiny, ty mají různé oprávnění, aby různí uživatelé byli kontrolováni, jaké části systému smí využívat.

Systém je vyvíjen firmou Consulta Bürotechnik s.r.o., která provádí i servis, instalaci a uvedení systému do provozu včetně zaškolení personálu. V nabídce firmy je také hardware, jako jsou pokladny či scannery, který je vhodný pro použití se systémem. Společnost nabízí systém ke zkušebnímu použití zdarma na dobu 30 dní. Je prodáván v několika verzích, základní verzi lze pořídit za 8.349 Kč, úplnou verzi za 13.189 Kč. K zakoupení je i velké

množství rozšiřujících modulů (např. modul statistik prodejů, modul pro rozšířené obchodní funkce ad.), ze kterých si zákazník může podle potřeby vybrat [17].



obr. 3-1: Hlavní obrazovka prodejního systému [17]

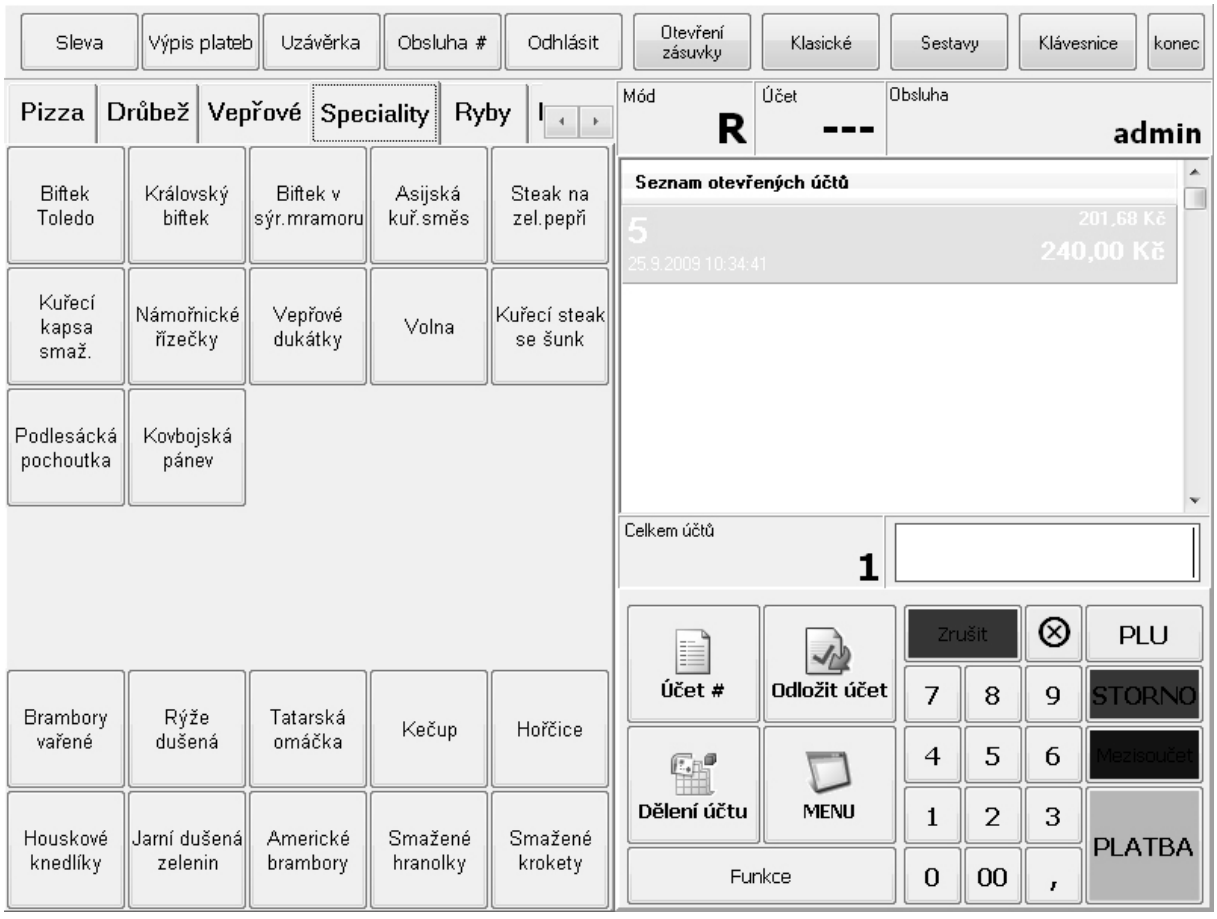
3.2 TPOS

TPOS je systém podobný předchozímu, avšak s jednodušším ovládáním pro uživatele, ale chudší na množství funkcí. Systém se zaměřuje zejména na hotelovou a restaurační oblast, avšak verze maloobchod jej umožňuje použít i v maloobchodních prodejnách. Systém funguje na principu klient/server.

Programový modul Terminál, který je součástí systému, je plnohodnotnou náhradou registrační pokladny. V rámci systému funguje jako klientský program. V rámci tohoto modulu se provádí jednotlivé prodeje, tisknou účtenky a provádějí další prodejní operace. Grafické uživatelské rozhraní je nastavitelné podle potřeb. Terminál dokáže komunikovat s různými periferními zařízeními, např. tiskárny, scannery čárových kódů apod.

Programový modul Maloobchod je jádrem systému, pomocí něj se spravují jednotlivé klientské stanice v rámci sítě. Modul obsahuje skladovou evidenci a funkce pro pohyb zboží, statistické a analytické nástroje a umožňuje tisk různých sestav, např. statistických ohledně prodejů, pohybu zboží, inventury ad.

Na oficiálních stránkách je možné zdarma stáhnout modul Terminál, ovšem modul maloobchod k vyzkoušení stáhnout nelze. Ceník systému není na oficiálních stránkách k dispozici. Z těchto stránek není jasné, zda jsou k systému poskytovány doplňkové služby jako instalace systému, zavedení do provozu nebo uživatelská podpora. K dispozici není ani náhled, jak vypadá modul Maloobchod, pouze náhled modulu Restaurace (viz obr. 3-2), který dle dostupných informací je modulu Maloobchod podobný [18].



obr. 3-2: Modul restaurace systému TPOS [18]

3.3 AWIS Obchod

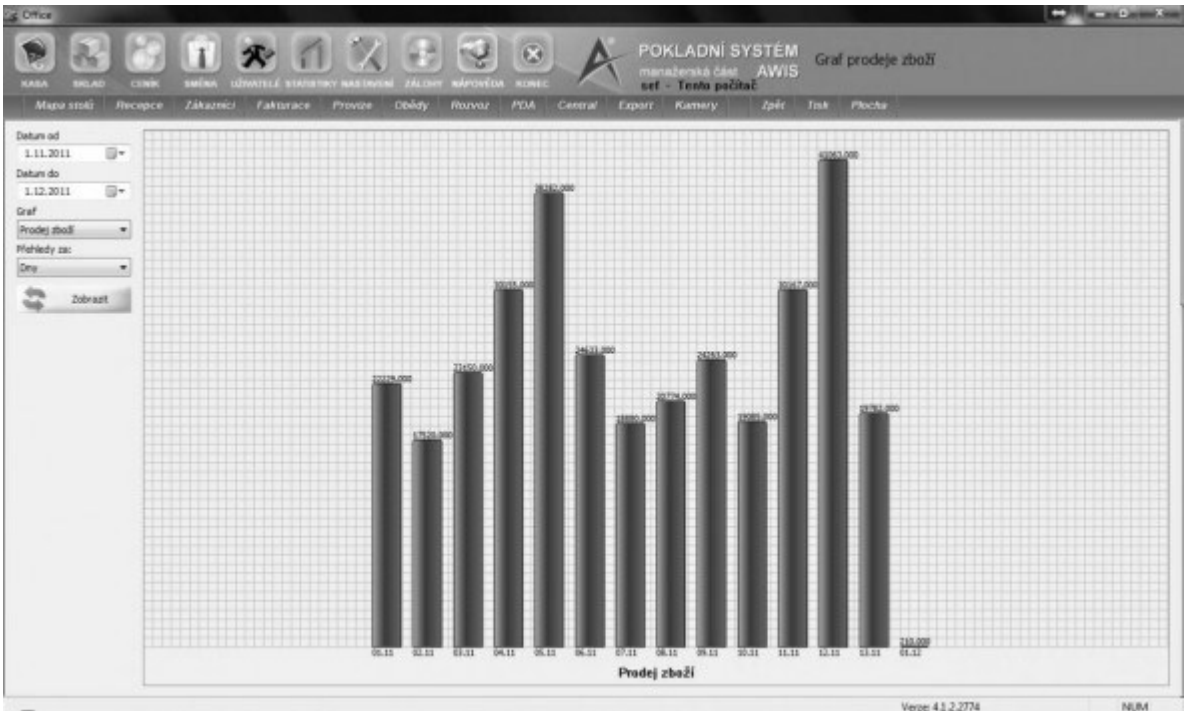
Pokladní systém AWIS Obchod je určený zejména pro maloobchody a velkoobchody a je vhodný pro obchody s různým zaměřením svého sortimentu. Součástí systému je i skladová evidence. Systém je velice propracovaný a oproti předchozím nabízí mnohem více využitelných funkcí. Hlavní předností je, že je možné systém použít nejen v jednom obchodě, ale v celé prodejní síti (ve verzi AWIS – Obchodní síť). Pro každého uživatele systému se vytváří uživatelský účet, kterému jsou přiřazena různá oprávnění pro práci se systémem.

Z oficiálního popisu systému na stránkách společnosti, která systém vyvíjí a dodává, lze usoudit, že systém funguje na principu klient/server architektury. Systém na klientských stanicích umožňuje použití periferních zařízení (scannery čárových kódů, tiskárny, váhy, bankovní terminály), provádění prodejních operací nebo platby v různých měnách.

obr. 3-3: Kasa - část systému používaného obsluhou [19]

Oproti předchozím systémům je možné propojit i s různými ekonomickými systémy pro daňovou evidenci a vedení účetnictví. Systém tedy dokáže pokrýt mnoho oblastí jako je fakturace, skladové evidence, mzdové účetnictví, podvojné účetnictví ad. Obsaženy jsou také analytické a statistické nástroje, které umožňují vytvořit širokou škálu tiskových sestav včetně grafických reportů (viz obr. 3-4) [19].

Systém je dodáván a vyvíjen firmou A.W.I.S. Správa, systémy s.r.o., která na tomto trhu funguje již od roku 1992. Systém je dodáván spolu s hardwarem a v základní verzi, která obsahuje software a jednu pokladnu, je jeho cena 15.609 Kč. Řešení, které pokryje potřeby např. prodejny s potravinami, je dostupné za 35.400 Kč bez DPH a jeho součástí je 15" LCD monitor, PC s operačním systémem, pokladní zásuvka, zákaznický display, scanner čárových kódů, pokladní tiskárna, váha a samotný systém AWIS Obchod. V ceně systému je zahrnuto i zaškolení personálu. Společnost navíc umožňuje upravení systému podle požadavků objednatele. Na stránkách společnosti je také možné stáhnout zkušební verzi.



obr. 3-4: Ukázka grafu prodeje zboží [19]

3.4 Shrnutí

Na trhu je v dnešní době mnoho podobných pokladních a skladových systémů, které jsou do značné míry velice propracované, a které jsou dále vyvíjeny. Systém, který je navržen a vyvíjen v rámci této bakalářské práce, se s těmi profesionálními nedá pořádně srovnávat, jelikož nenabízí tak pokročilé funkce. Na druhou stranu jsou tato řešení pro menší podniky nákladná a poskytují zbytečné funkcionality, které by zůstaly nevyužity [19].

4 NÁVRH A IMPLEMENTACE SYSTÉMU

Systém byl vytvořen pro maloobchod, který se zaměřuje na prodej potravin. Do doby než byl zaveden systém, bylo zde použito jedné pokladny, kde její obsluha zadávala pouze cenu prodávaného výrobku a pokladna tiskla účtenky bez jeho jednoznačné identifikace. Skladová evidence byla vedena pouze v papírové podobě.

Zadáním bylo vytvořit skladový a prodejní (resp. pokladní) systém, který by zjednodušil práci zaměstnanců.

4.1 Specifikace požadavků

Před samotným návrhem a vytvořením systému bylo nutné definovat požadavky na samotný systém. Ty byly stanoveny na základě zadání a rozhovorů s budoucími uživateli systému.

4.1.1 Funkční požadavky

V první řadě bylo nutné stanovit funkční požadavky na systém, tedy požadavky, které popisují jednotlivé funkce systému a to jak se má systém chovat.

Základních požadavkem na prodejní část bylo to, aby systém vedl evidenci prodejních aktivit, zejména tvorba, tisk a ukládání účtenek, tržby jednotlivých zaměstnanců v jednotlivých dnech a uskutečnění prodejů v návaznosti na skladovou evidenci. Začátkem je přihlášení a autorizace zaměstnance k prodejní části systému a jeho funkcím, zejména k prodeji vybraného zboží zákazníkům. Běžnou formou, jak je zboží vybráno v systému k prodeji, je použití scanneru čárových kódů, pokud zboží tento kód nemá, je třeba projít seznam zboží, ve kterém se provede jeho výběr. V průběhu tohoto procesu je nutné, aby bylo možné jednotlivé položky prodeje mazat či upravovat jejich množství. Nakonec bude vytištěn doklad o zaplacení, který se zároveň uloží do systému, ze skladové evidence se odečte prodané zboží a aktualizují se informace o tržbě a stavu hotovosti v pokladně.

Dalšími požadavky na prodejní část systému je, aby kontroloval aktuální stav hotovosti v pokladně, možnost vložení či odebrání hotovosti (např. před otevírací dobou vložit počáteční hotovost) či zobrazení aktuální tržby.

Hlavním požadavkem na skladovou část systému je základní vedení skladového hospodářství. Uživatel musí mít možnost nahlížet na zásoby jednotlivých zboží a možnost upravovat stav zboží při jeho dodání nebo inventarizaci.

4.1.2 Nefunkční požadavky

Dále byly stanoveny nefunkční požadavky. Ty přímo nesouvisí s chováním systému, ale dávají systému omezující podmínky.

- Grafické uživatelské rozhraní bude přehledné a jednoduše ovladatelné. Pro jeho obsluhu bude využito klávesnice a myši. Při manipulaci se zbožím bude využito scanneru čárových kódů.

- Prodejní a skladová část systému budou pracovat na dvou zařízeních a jejich funkce budou odděleny, avšak je nutné, aby pracovaly zároveň nad jednou datovou základnou.

- Bez autentizace uživatele nebude možné pracovat se systémem.

- Skladová část systému bude používaná pouze pověřenou osobou, tedy ne všichni zaměstnanci zde budou mít přístup a zároveň vedoucí pracovník zde bude provádět správu uživatelských účtů a spravovat seznam zboží.

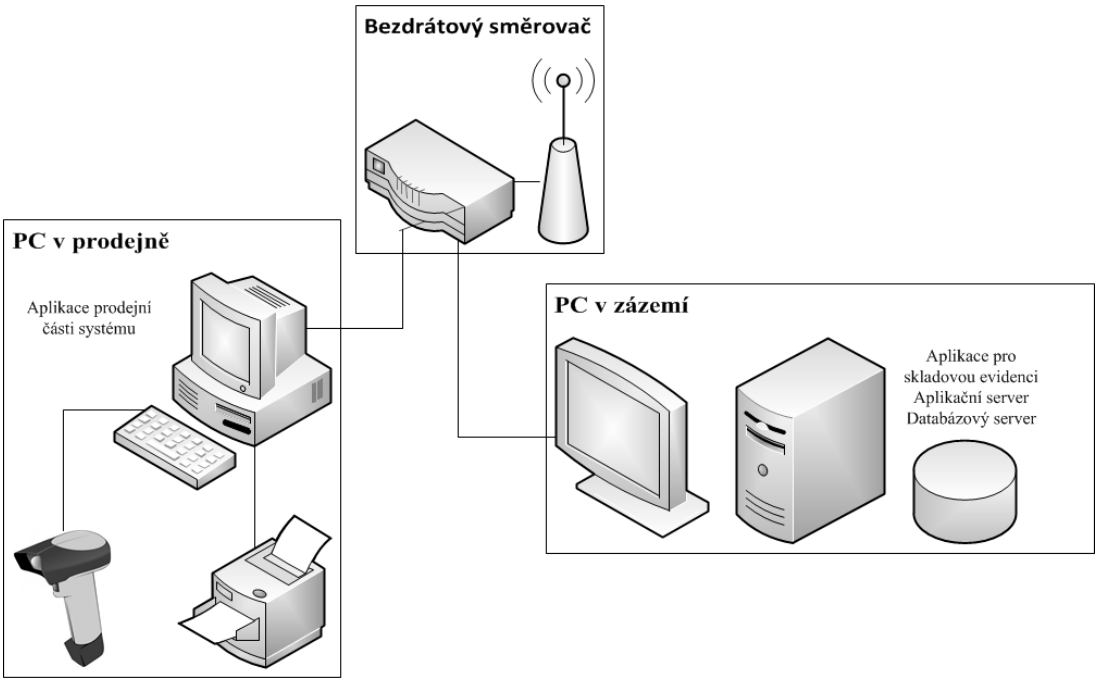
4.2 Návrh systému

S přihlédnutím na požadavky bylo nutné navrhnout systém z různých pohledů. Bylo nutné popsat budoucí architekturu systému, jeho datovou základnu a to, jak se systém bude chovat v určitých situacích.

4.2.1 Fyzická architektura

Vzhledem k požadavku na skladovou a prodejní část systému, kdy obě části mají pracovat nezávisle na sobě, bylo rozhodnuto, že se použijí dvě PC. Každé PC bude umístěno na jiném místě, jedno přímo v prodejně a druhé v zázemí. Na PC umístěném v zázemí bude běžet aplikační a databázový server, zároveň bude sloužit jako klient pro skladovou část systému, a tím se umožní případné budoucí přesunutí aplikace pro skladovou evidenci na jiné

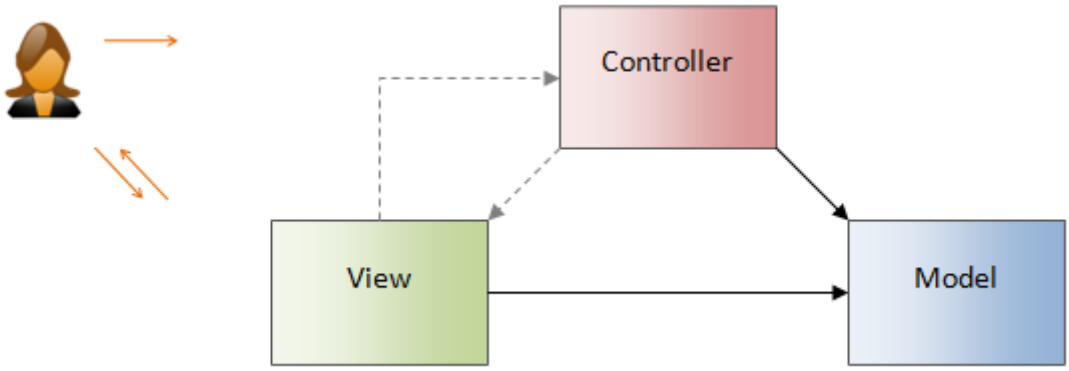
PC. PC umístěné v prodejně bude sloužit jako pokladna a bude na něm používána aplikace pro prodejní část systému. Komunikace mezi oběma PC bude zabezpečena pomocí lokální počítačové sítě (LAN). Jelikož vzdálenost mezi oběma PC a jejich umístění nedovoluje požití kabeláže pro jejich propojení, bude použito bezdrátového směrovače. Architektura je zachycena na obr. 4-1, ze kterého je patrné, že současně je použita třívrstvá architektura.



obr. 4-1: Fyzická architektura

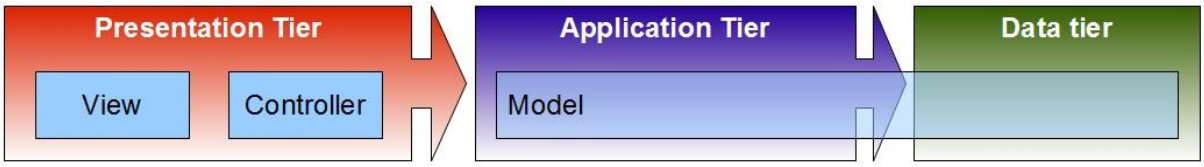
4.2.2 MVC Architektura

MVC je zkratka tří anglických slov, Model (model), View (pohled) a Controller (řadič). Účelem MVC je logické rozdělení jednotlivých tříd do výše uvedených skupin, přičemž jednotlivé skupiny spolu komunikují (viz obr. 4-3), avšak mají na sebe jen minimální vliv. Skupina view, jak je z názvu patrné, je ta, která je nejblíže uživateli a je zde obsaženo grafické uživatelské rozhraní. Model představuje třídy, které se starají o data a manipulaci s nimi. V případě navrhovaného systému tyto třídy zabezpečují komunikaci oběma směry k databázi. Poslední skupinou je controller, ten je pojítkem mezi předchozími skupinami a reaguje na události, nejčastěji pocházejícími od uživatele, a provádí změny v modelu nebo pohledu [20].



obr. 4-3: Vztahy mezi komponentami MVC a uživatelem [20]

Vzhledem k tomu, že je zde využita také třívrstvá softwarová architektura (viz předchozí kapitola), je nutné se zmínit i o tom, že tyto dvě architektury spolu souvisí. Mnohdy bývají považovány za stejné, avšak tomu tak není, jejich vztah je ilustrován na následujícím obrázku [21].



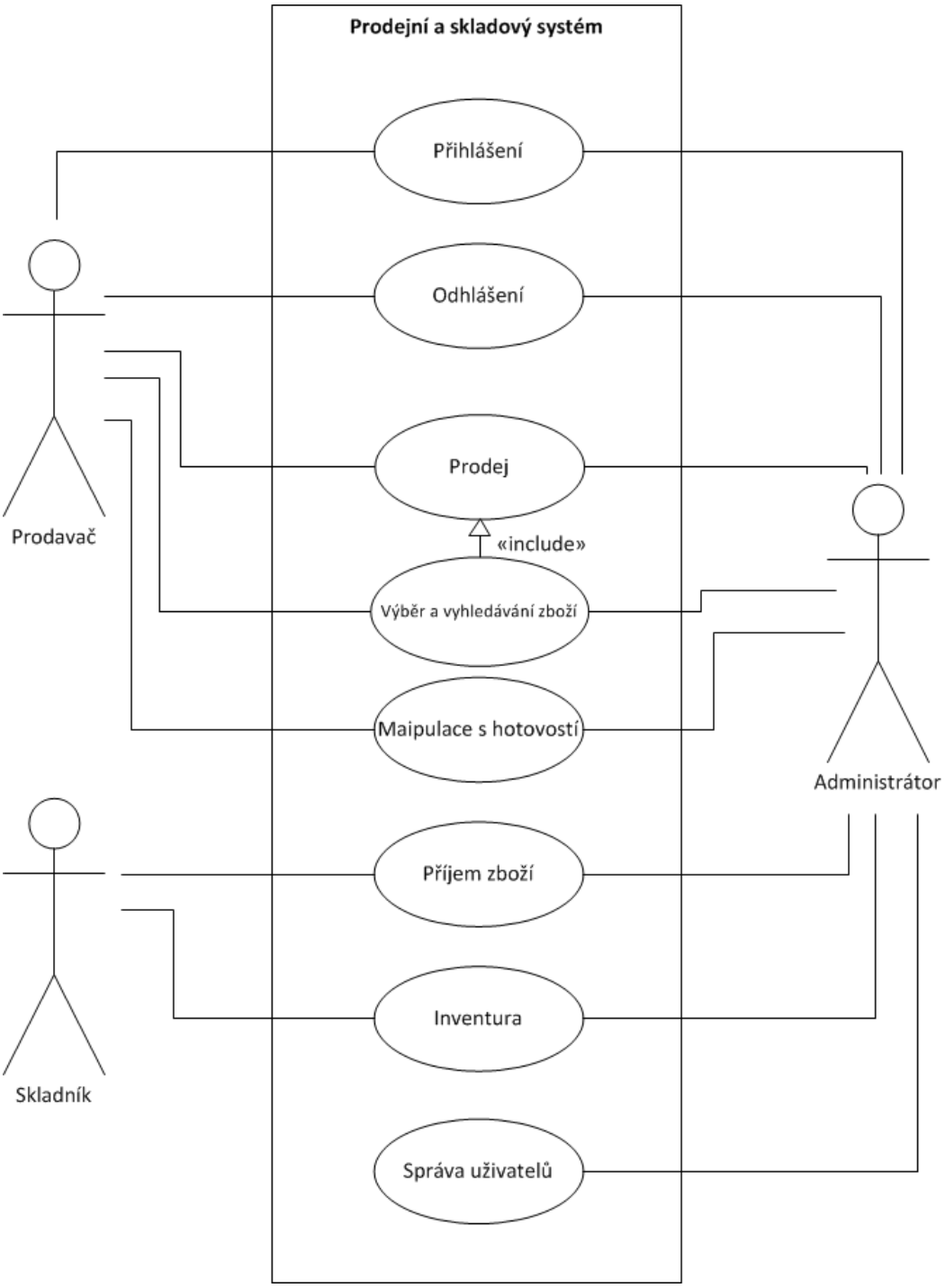
obr. 4-2: Vztah třívrstvé a MVC architektury [21]

4.2.3 Use case – případy užití

Případy užití jsou funkce systému, které jsou využívány aktéry, uživateli systému. Každý případ užití představuje právě jednu funkčnost a popisuje jakým způsobem a za jakých podmínek se bude systém chovat při interakci s aktérem. Případy užití na sebe mohou určitým způsobem na sebe navazovat. Tyto vazby a jednotlivé případy užití jsou dále popsány. V rámci systému vystupuje několik aktérů, a to prodavač, skladník a administrátor. Prodavač je uživatelem pokladní části sytému, skladník skladové části a administrátor má přístup k celému systému. Na obr. 4-4 je poté diagram případů užití, který zobrazuje interakce mezi aktéry a systémem [22].

Přihlášení a odhlášení – přihlášení je prvním případem užití jak skladové, tak prodejní části systému a zahajuje se jím interakce se systémem. Bez úspěšné autentizace a autorizace uživatele není možné do systému vstoupit. Odhlášení je poslední aktivitou, kterou aktér provádí se systémem, ukončují se jím veškeré interakce mezi aktérem a systémem. U prodejní části se přihlášením tzv. otevírá pokladna a zaznamenávají se aktivity aktéra, při odhlášením se naopak pokladna uzavírá.

Prodej – je nejčastějším případem užití prodejní části. V rámci prodeje aktér sestavuje seznam prodávaného zboží pomocí scanneru čárových kódů nebo vyhledáním zboží v systému. Poté je případně seznam dále upravován, položky mohou být mazány anebo měněno jejich množství. Prodej je ukončen příjmem hotovosti a vystavením pokladního dokladu.



obr. 4-4: Diagram případů užití

Výběr a vyhledávání zboží – i když je výběr a vyhledávání zboží součástí případu užití prodej, je občas nutné vyhledat či vybrat zboží mimo tento případ, a proto je uveden ještě samostatně. Dalším důvodem je, že se vztahuje i na skladovou část systému. Příkladem, kdy je nutné použít tento případ samostatně, je potřeba zjistit, zda je zboží na skladě nebo jaká je jeho cena.

Manipulace s hotovostí – prodavač může manipulovat s hotovostí v pokladně, např. před otevřením, když vkládá do pokladny počáteční hotovost. Aby bylo možné sledovat aktuální stav hotovosti v pokladně, je nutné, aby bylo možné provádět takové operace v rámci prodejního systému.

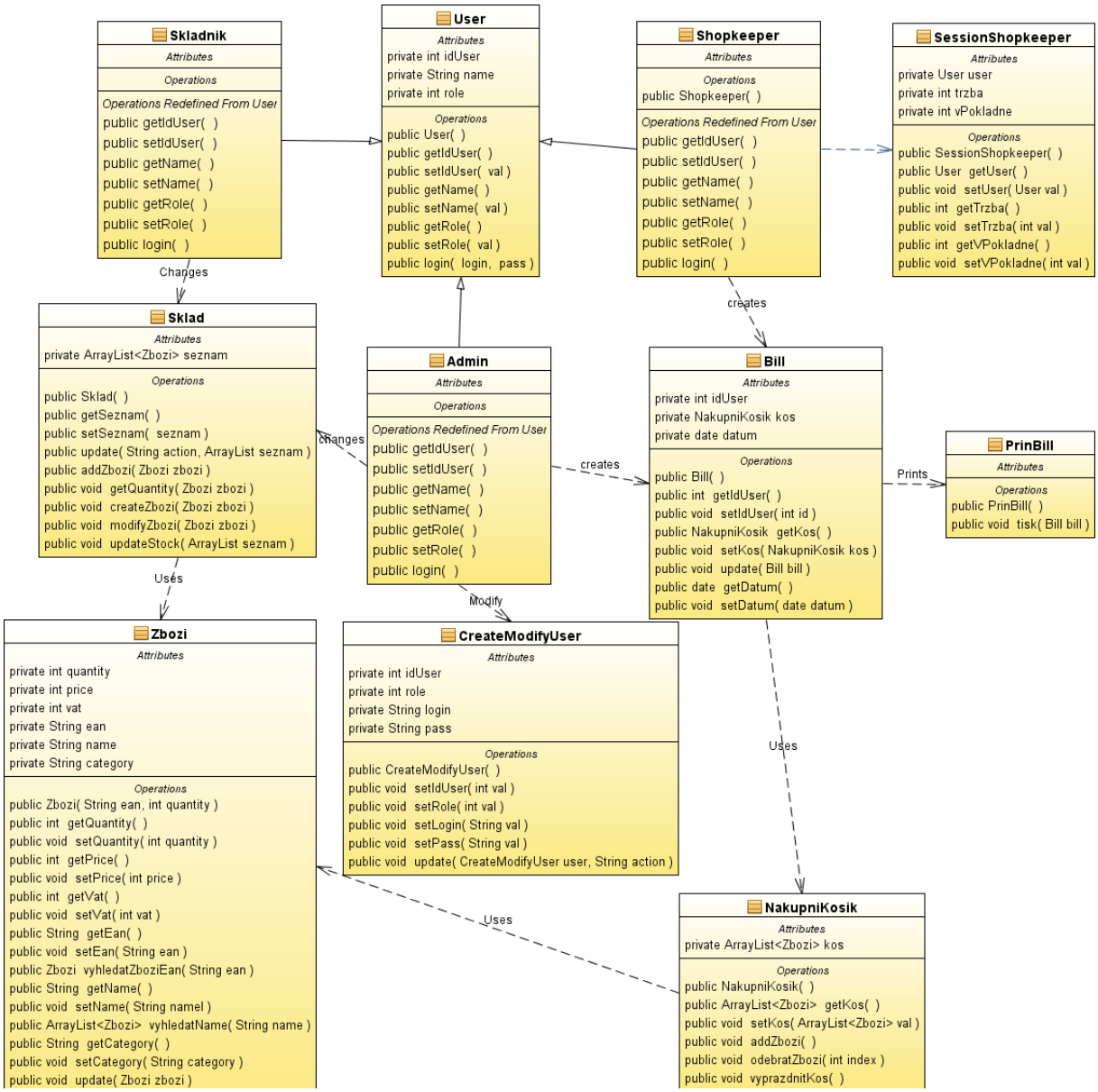
Příjem zboží – je případem užití skladové části. Aktér, skladník, přijme zboží od dodavatele, umístí je ve skladu a do systému musí zadat kolik a jakého zboží bylo přijato.

Inventarizace – je dalším případem užití skladové části. V první řadě musí aktér provést inventuru, přepočítat fyzicky zboží na skladě, a poté tyto skutečnosti porovnat se stavem zboží v systému. V případě nesrovnalostí je nutné data v systému opravit.

Správa uživatelů – je posledním případem užití. Pouze administrátor smí vytvářet nové nebo mazat staré uživatele a spravovat jejich přístupová práva do jednotlivých částí systému.

4.2.4 Diagram tříd

Diagram tříd je statickým pohledem na modelovaný systém. Pomocí něj jsou zobrazeny typy objektů systému a jejich vztahy. Diagram zobrazuje strukturu tříd, avšak ne jejich interakce v čase. Základním prvkem je třída, ta je popsána názvem, jednotlivými atributy a metodami. Na obr. 4-5 je zachycen diagram tříd serverové části systému [4].



obr. 4-5: Diagram tříd serverové komponenty

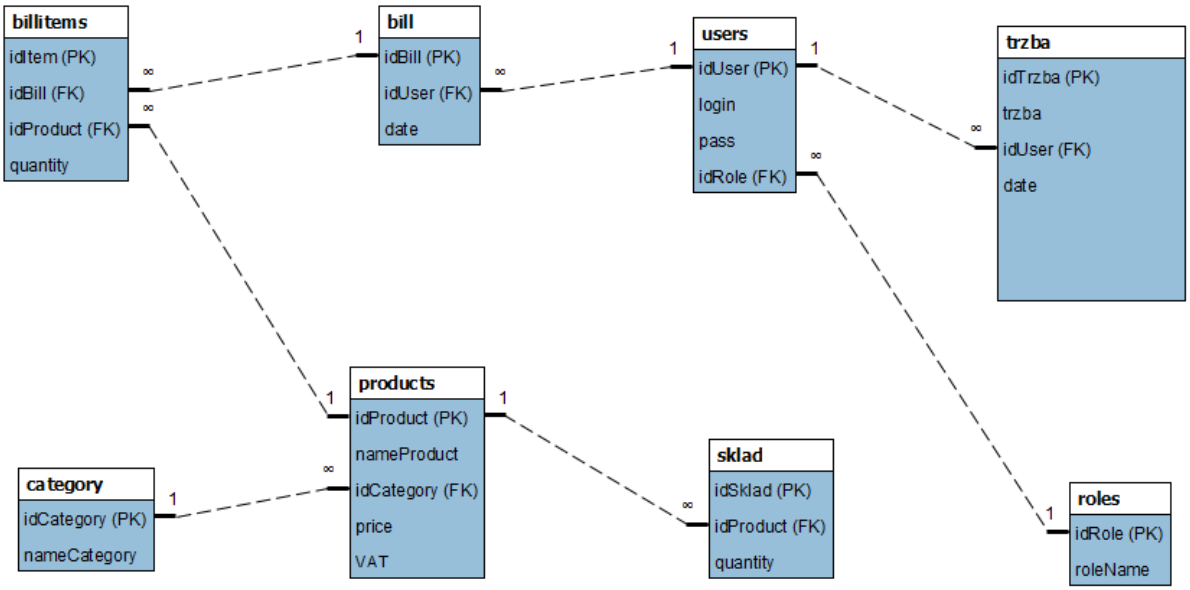
4.2.5 Návrh datové struktury

V rámci systému je nutné určit, jakým způsobem budou ukládána, uchovávána a poskytována data. Prvním krokem bylo určit, jaká data se budou používat a podle toho rozhodnout jaký datový model využít. V minulosti bylo využíváno hierarchické a později síťové struktury dat. V rámci tohoto systému byl vybrán relační databázový model, který se v dnešních databázích hojně využívá.

S relační databází a jejím návrhem souvisí pojem entitně-relační diagram (ERD – Entity–realationship diagram). Ten slouží právě k návrhu databáze, k zobrazení entit a relací mezi nimi. Entitou se v tomto případě rozumí nějaký objekt nebo skutečnost reálného světa a

jeho vlastnosti. V databázi představuje jednu tabulku a její atributy. Příkladem entity, která je zároveň obsažena v navrhovaném systému, je uživatel (atributy – přihlašovací jméno, heslo, identifikační číslo, role v systému) nebo produkt (atributy – název, cena ad.). Relace představuje vztah mezi jednotlivými entitami. U jednotlivých relací poté určujeme kardinalitu vztahu, která vyjadřuje, kolik výskytů jedné entity může vytvořit vztah s kolika výskyty druhé entity. Rozlišujeme vztah 1:1, 1:N a M:N [4].

Na obr. 4-6 je ER diagram navrhované databáze. Jsou zde všechny entity, jejich atributy a vazby mezi nimi.



obr. 4-6: ER diagram databáze prodejního a skladového systému

4.3 Implementace

Dalším krokem byla implementace systému. V této podkapitole jsou stručně popsány použité nástroje a poté samotný postup implementace.

4.3.1 MySQL

MySQL je databázový systém vyvíjený společností Oracle Corporation. Jedná se o multiplatformní systém, je možné jej provozovat nad různými operačními systémy, lze jej nainstalovat na MS Windows, Linux, Mac OS a další operační systémy. MySQL je open-source software, což znamená, že zdrojový kód je dostupný všem a při dodržení určitých

podmínek je možné jej využívat. V dnešní době se jedná o jeden z nejpopulárnějších open-source databázových systémů.

Jak je z názvu patrné, je zde využit jazyk SQL (Structured query language – Strukturovaný dotazovací jazyk), který se používá pro práci s daty. MySQL disponuje vlastnostmi, které jsou potřebné pro vyvíjený systém, těmi jsou např. uložené procedury, triggery, pohledy a spolu s uložištěm dat InnoDB také transakce. Zejména multiplatformita a tyto vlastnosti jsou důvodem, proč byl vybrán databázový systém MySQL [23].

4.3.2 Java

Java je objektově orientovaný programovací jazyk vyvinutý firmou Sun Microsystems, nyní vlastněný společností Oracle Corporation, a patří mezi nejoblíbenější a nejpoužívanější programovací jazyky vůbec. Obdobně jako MySQL patří Java mezi open-source software a je taktéž multiplatformní.

Multiplatformity je v Javě dosaženo díky Java Virtual Machine (JVM), které funguje jako virtuální stroj a slouží ke spuštění programového kódu. K Javě je poskytováno mnoho nástrojů a knihoven, které umožňují tvorbu jak jednoduchých programů, tak rozsáhlých systémů. Zde jsou použity zejména knihovna Swing a serverové komponenty Enterprise Java Beans (EJB) [24].

a) Swing

Swing je knihovna uživatelských prvků, které umožňují ovládání počítače pomocí grafického rozhraní. Je součástí frameworku JFC (Java Foundation Classes), které slouží právě pro tvorbu grafického uživatelského rozhraní. Pomocí swingu je možné vytvářet komponenty uživatelského rozhraní, jako jsou textová pole, rozbalovací seznamy, tlačítka, dialogy, přepínače apod. Všechny komponenty uplatňují „lightweight“ přístup (lehký přístup), což znamená, že jejich vzhled není závislý na operačním systému. Pro zjednodušení tvorby je navíc použit Swing GUI Builder, který je součástí NetBeans IDE (viz dále) [25].

b) Enterprise JavaBeans

Technologie Enterprise JavaBeans (EJB) jsou řízené serverové komponenty na platformě Java Enterprise Edition (Java EE) a zabezpečují aplikační vrstvu softwarové architektury a jejich cílem je právě oddělit aplikační vrstvu od prezentační. EJB jsou určeny pro rychlou a jednoduchou tvorbu podnikových systémů a jejich nespornou výhodou je znovu použitelný kód, bezpečnost nebo jednodušší testování a integrace [26].

Knihovna Swing a EJB dohromady představují MVC architekturu, přičemž Swing patří do skupin view a controller a EJB do skupiny model.

4.3.3 NetBeans IDE

NetBeans IDE (Integrated Development Environment – vývojové prostředí) je open-source nástroj, pomocí kterého programátoři mohou vytvářet, psát, překládat, ladit a šířit aplikace a programy. Toto vývojové prostředí je vytvořeno v jazyce Java, avšak je možné jej použít s téměř jakýmkoliv programovacím jazykem a je možné je používat zdarma i v komerční sféře. NetBeans původně začal jako studentský projekt v České republice [27].

4.3.4 GlassFish

GlassFish je aplikační server vyvinutý společností Sun Microsystems pro platformu Java Enterprise Edition. Stejně jako výše zmíněné produkty a nástroje patří tento server mezi open-source software. GlassFish je referenční implementací, která podporuje Enteprise JavaBeans, servlety, JavaServer Pages, JavaServer Faces a další. Slouží především jako prostředí pro nasazení a spouštění podnikových aplikací v počítačové síti [28].

4.3.5 Postup implementace

Jelikož neexistoval žádný základ, do kterého by se prodejní a skladový systém integroval, bylo nutné začít od začátku. Prvním krokem bylo vytvoření místní počítačové sítě. Jedinou investicí bylo pořízení bezdrátového směrovače, jelikož potřebné počítače již zde byly. Poté se přistoupilo k instalaci nutného vybavení, zejména MySQL databázového systému a aplikačního serveru Glassfish.

Dále byla vytvořena v databázovém systému relační databáze „obchod“ podle návrhu a ER diagramu (viz kapitola 4.2.5 Návrh datové struktury). Postupně byly vytvořeny tabulky databáze, jejich sloupce a jednotlivé relace mezi tabulkami. Na obr. 4-7 je ukázán kód vytvářející tabulku s účtenkami, obdobně byly vytvořeny i tabulky ostatní.

```
1  CREATE  TABLE `obchod`.`bill` (
2    `idbill` INT NOT NULL ,
3    `idUser` INT NOT NULL ,
4    `date` DATE NOT NULL ,
5    PRIMARY KEY (`idbill`) ,
6    INDEX `FK_User_idx` (`idUser` ASC) ,
7    CONSTRAINT `FK_User`
8      FOREIGN KEY (`idUser` )
9      REFERENCES `obchod`.`users` (`idUser` )
10     ON DELETE NO ACTION
11     ON UPDATE NO ACTION);
12
```

obr. 4-7: Příklad kódu vytvářejícího tabulku databáze

Další krok představoval vytvoření EJB jako řídícího prvku jak skladové, tak prodejní části. Jelikož v systému budou klientské aplikace přistupovat k databázi současně, bylo nutné ošetřit to, aby se nestal případ, kdy obě aplikace najednou přistupují k databázi a přepisují stejná data. Současně s EJB bylo vytvořeno rozhraní s anotací `@Remote`, což umožňuje volání metod ze vzdáleného JVM. EJB implementuje právě toto rozhraní s tím, že musí přepsat jeho metody. V rámci EJB jsou části kódu, přistupujících k databázi, nejčastějším způsobem přístupu pak je volání uložených procedur.

```
@Remote
public interface MySessionRemote {

    String getResult();

    String login(String login, String pass);

    void addZbozi(String ean, int quantity);

    void vyprazdnitKos();

    int getTrzba();
```

obr. 4-8: Fragment programového kódu vzdáleného rozhraní

Součástí EJB je i třída PrintBill vytvářející účtenky, které se budou tisknout pokladní tiskárnou. Třída implementuje rozhraní Printable, které toto umožňuje. Tiskárna na šířku tiskne 42 znaků, a proto bylo nutné v této třídě upravit případné delší názvy produktů tak, že část textu byla smazána. Účtenka se skládá ze záhlaví, kde jsou základní údaje o prodejně, obsahu, kde jsou jednotlivé prodané položky, celková suma, datum prodeje, číslo účtenky apod. a zápatí, kde jsou doplňující informace. Ukázka účtenky je na obr. 4-9.

```
                    NAZEV FIRMY

                Ulice 25/17a, XXX XX Mesto
              ICO: 12345678, DIC: CZ123456789

Danovy doklad                          15.03.2013
0000000001

------------------------------------------

Polozka                      Mnozstvi    Cena
------------------------------------------

Rama Classic 500g                 1     36.00
Delisa or.33g                     2     18.00
Tatar. om. agricol 250ml          1     22.00
------------------------------------------
Hotovost:                               76.00
------------------------------------------
Sazba DPH                   Dan        Celkem
DPH 15%                    9.91         66.09
DPH 21%                    0.00          0.00
------------------------------------------
Celkem                     9.91         66.09
------------------------------------------
                DEKUJEME ZA NAKUP
------------------------------------------
```

obr. 4-9: Příklad pokladního dokladu

Jakmile bylo vytvořeno jádro obou aplikací, bylo pomocí knihovny Swing vytvořeno grafické uživatelské rozhraní, které uživatelům umožňuje interakci se systémem a v rámci celého systému představuje klienta. Pro skladovou i prodejní část bylo vytvořeno samostatné uživatelské rozhraní, jediné v čem obě jsou stejné, je počáteční přihlašovací okno. Zde je implementováno rozhraní `ActionListener`, které umožňuje „naslouchat“ a pokud nastane v kódu definovaná událost, je volána metoda `actionPerformed (ActionEvent e)` a jsou provedeny příkazy této metody. Díky tomuto je možné např. při stisku klávesy Enter v textovém poli simulovat stisk tlačítka (příklad této přepsané metody je na obr. 4-10). Dalším příkladem, kdy bylo využito tohoto rozhraní, je snímání čárového kódu zboží. Výhodou je, že scanner funguje podobně jako klávesnice, odesílá znaky ze vstupu, tak jako by byly psány klávesnicí.

Poté co byly klientské aplikace hotovy, přistoupilo se k zadávání dat do databáze. Jelikož veškerou administraci zastává skladová část, tedy vložení zboží do databáze, množství zboží na skladě a v případě přihlášeného administrátora, jehož uživatelský účet byl do databáze zadán samostatně, i seznam uživatelů.

```
    @Override
public void actionPerformed(ActionEvent e) {
    String cmd = e.getActionCommand();
    if (cmd.equals("Open") || cmd.equals("enter")) {
        char[] input = txtPass.getPassword();
        String heslo = new String(input);
        if (txtLogin.getText().equals("") || heslo.equals("")) {
            JOptionPane.showMessageDialog(rootPane, "Pole nesmi byt prazdne");
        } else if (mySession.login(txtLogin.getText(), heslo).equals("ano")) {

            dispose();
            MainWindow afterLogin = new MainWindow();
            afterLogin.setVisible(true);
        } else {
            JOptionPane.showMessageDialog(rootPane,
                    mySession.login(txtLogin.getText(), heslo));
        }
    }
```

obr. 4-10: Přepsaná metoda actionPerformed definující události při přihlašování

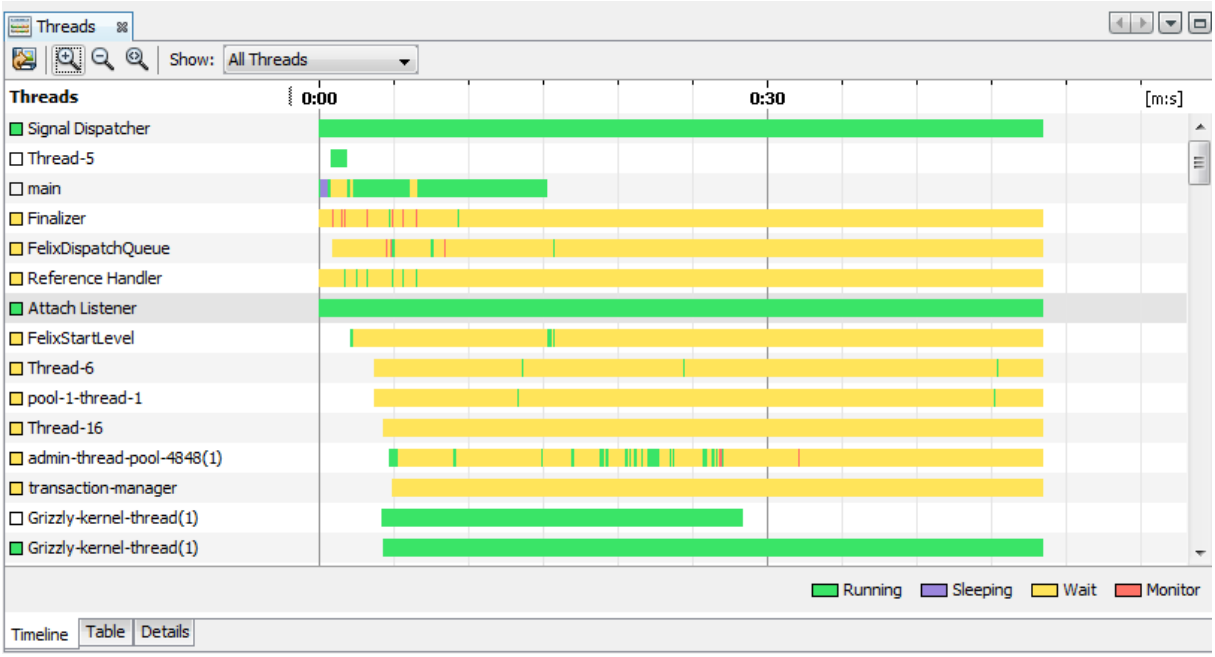
4.4 Testování

Testování systému probíhalo několika způsoby a to během a po implementaci. Navíc správnost implementovaných funkcí je zde důležitější, vzhledem k tomu, že systém pracuje i s penězi.

Testování během implementace znamenalo odzkoušet jednotlivé metody, zda pracují požadovaným způsobem a zda vykonávají potřebné funkce. Při nalezení chyby bylo nutné najít v kódu chybu a opravit ji. K tomuto testování byla vytvořena testovací databáze, která byla vytvořena podle zamýšlené databáze.

Testování po implementaci probíhalo za provozu souběžně se starým způsobem vedení prodejny (simuloval se ostrý provoz), jelikož se tak zabránilo tomu, aby nenalezené a neopravené chyby systému měly dopad např. na administraci nebo samotný prodej. V této fázi se zejména testovala komunikace mezi databází a aplikacemi a samotné fungování systému jako celku. Zkoušely se i rozličné neobvyklé uživatelské vstupy, které by mohly

Třetí způsob probíhal pomocí nástroje NetBeans Profiler, který je součástí vývojového prostředí. Ten umožňuje sledovat různé ukazatele při běhu aplikace, ze kterých se dá posoudit její kvalita a pomocí kterých lze nevhodné části kódu. Lze také sledovat využití paměti nebo procesoru.



obr. 4-11: Použití NetBeans Profiler

5 ZÁVĚR

Dobře navržený a implementovaný informační systém poskytuje jeho vlastníkovi velkou konkurenční výhodu, jelikož je prostředkem pro snižování nákladů, a to jak časových, tak finančních.

Základní pojmy a teoretické znalosti, které se vztahují k informačním systémům, jsou uvedeny v druhé kapitole. Je zde popsáno, co to vlastně informační systémy jsou, co je jejich účelem, z čeho se skládají a jak se mohou dělit. Dále jsou zde uvedeny možnosti, jak je lze získat. V posledních částech kapitoly o teoretických základech informačních systémů je popsáno, že existují různé, až protichůdné přístupy k jejich tvorbě a jsou zde popsány jednotlivé metodiky vývoje. Nakonec je popsáno, které metodiky pak jsou v dnešní době nejvíce využívány.

Ve třetí kapitole byl zkoumán český trh s prodejními, pokladními a skladovými systémy. Jsou zde popsány některé produkty různých společností, které poskytují rozličné množství funkcí jejich uživatelům. Kromě zde popsaných však existuje ještě celá řada systémů, které poskytují stejné funkcionality nebo jsou si na první pohled velmi podobné, že zde nejsou popsány. Z této kapitoly je zřejmé, že na českém trhu je možné získat téměř dokonalý systém, který se stále vyvíjí a vylepšuje, avšak velmi malé prodejny a obchody, které by takový systém zakoupily, by spolu s ním zakoupily nespočet funkcí, kterých by nevyužily, což by se dalo považovat za plýtvání penězi. Systém, který je popsán ve čtvrté kapitole, se s komerčními systémy na českém trhu konkurenčně sice nemůže rovnat, avšak pro potřeby, pro které byl navržen, je plně dostačující.

Cílem této bakalářské práce bylo navrhnout a implementovat prodejní a skladový systém, čemuž se věnuje čtvrtá kapitola. Ta popisuje, jak byl tento systém, který poskytuje malé prodejně právě tolik funkcí, kterých využije, navržen a implementován. Celý tento proces byl rozdělen do dílčích cílů. V prvé řadě bylo nutné analyzovat stávající systém, protože v některých případech je možné využít jeho části. Jelikož zde žádný informační systém nebyl, přistoupilo se přímo ke specifikaci požadavků, tím se zabývá první podkapitola. Spolu se zadavatelem a jeho zaměstnanci byly definovány funkční a nefunkční požadavky. Bez specifikace by totiž nebylo možné pokračovat.

Třetím cílem byl návrh systému. Bylo nutné vzít v úvahu nejen jednotlivé požadavky, ale také např. vnitřní uspořádání místností prodejny. Z toho důvodu byla navržena nejprve fyzická architektura systému, přičemž bylo zároveň rozhodnuto o využití třívrstvé softwarové architektury a MVC architektury. Poté byly požadavky transformovány do diagramu případů užití. Ten sloužil pro popis chování systému a jednotlivé případy užití byly blíže rozepsány. Nakonec byl vytvořen návrh datové struktury systémy a diagram tříd.

Čtvrtý cíl představuje samotná implementace. Pomocí jednotlivých nástrojů, které jsou ve třetí podkapitole popsány, byla nejprve vytvořena relační databáze a poté jednotlivé programové komponenty. Jak během, tak po implementaci byly komponenty podrobovány testování. Po odstranění nalezených chyb byl systém uveden do provozu.

Jednotlivé dílčí cíle návrhu a implementace byly naplněny a systém je nyní provozuschopný. Jak bylo řečeno výše, systém disponuje pouze nejnutnějšími funkcionalitami, avšak do budoucna by bylo možné implementovat různé analytické a podpůrné nástroje, např. upozornění na možnost brzkého vyčerpání zásob určitého zboží v závislosti na průměrné prodejnosti. Systém by bylo možné také rozšířit nejen v rámci lokální počítačové sítě, ale také prostřednictvím internetu, což by bylo vhodné, pokud by existovalo více prodejen.

SEZNAM POUŽITÝCH ZDROJŮ

1. ŠMÍD, Vladimír. *Management informačního systému*. 2005 [cit. 2013-03-08]. Dostupné z: http://www.fi.muni.cz/~smid/managis.html

2. BUCHALCEVOVÁ, Alena. *Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky*. 1. vyd. Praha: Grada, 2005, 163 s. ISBN 80-247-1075-7.

3. BRUCKNER, Tomáš et al. *Tvorba informačních systémů: principy, metodiky, architektury*. 1. vyd. Praha: Grada, 2012, 357 s. Management v informační společnosti. ISBN 978-80-247-4153-6.

4. DOUCEK, Petr. *Řízení projektů informačních systémů*. 2., rozš. vyd. Praha: Professional Publishing, 2006, 180 s. ISBN 80-869-4617-7.

5. GÁLA, Libor, Jan POUR a Zuzana ŠEDIVÁ. *Podniková informatika*. 2., přeprac. a aktualiz. vyd. Praha: Grada, 2009, 496 s. Expert (Grada). ISBN 978-80-247-2615-1.

6. MOLNÁR, Zdeněk. *Moderní metody řízení informačních systémů*. 1. vyd. Praha: Grada, 1992, 352 s. ISBN 80-856-2307-2.

7. POUR, Jan. *Informační systémy a technologie*. Vyd. 1. Praha: Vysoká škola ekonomie a managementu, 2006, 492 s. Edice učebních textů. Informační systémy a technologie. ISBN 80-867-3003-4.

8. SEDLÁČEK, Václav. *Principy a modely řízení podnikové informatiky: studijní opora disciplíny*. Vyd. 1. Třebíč: Západomoravská vysoká škola Třebíč, 2010, 92 s. ISBN 978-808-7385-067.

9. SKLENÁK, Vilém. *Data, informace, znalosti a Internet*. Vyd. 1. V Praze: C.H. Beck, 2001, xvii, 507 s. C.H. Beck pro praxi. ISBN 80-717-9409-0.

10. SODOMKA, Petr a Hana KLČOVÁ. *Informační systémy v podnikové praxi*. 2. aktualiz. a rozš. vyd. Brno: Computer Press, 2010, 501 s. ISBN 978-80-251-2878-7.

11. CHLAPEK, Dušan, Václav ŘEPA a Iva STANOVSKÁ. *Analýza a návrh informačních systémů*. 1. vyd. Praha: Oeconomica, 2011. ISBN 978-80-245-1782-7.

12. BUCHALCEVOVÁ, Alena. *Metodiky budování informačních systémů*. Vyd. 1. Praha: Oeconomica, 2009, 205 s. Vysokoškolská učebnice (Oeconomica). ISBN 978-80-245-1540-3.

13. BECK, Kent et al. *Manifesto for Agile Software Development* [online]. 2001 [cit. 2013-04-03]. Dostupné z: http://www.agilemanifesto.org

14. VERSIONONE. 7th Annual State of Agile Development Survey [online]. VersionOne, Inc. 2013 [cit. 2013-04-08]. Dostupné z: http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf

15. KOCH, Miloš et al. *Management informačních systémů*. Vyd. 2., přeprac. Brno: Akademické nakladatelství CERM, 2010, 171 s. ISBN 978-80-214-4157-6.

16. ZELENÝ, Jindřich. Architektura IS a její úloha v systémové integraci. In: *Systems Integration 2001:9th Annual International Conference*.. Praha: Prague University of Economics, 2001, s. 357-364 [cit. 2013-04-08]. ISBN 80-245-0169-4. Dostupné z: http://si.vse.cz/archive/index.asp?volume=2001

17. CONSULTA BÜROTECHNIK. Pokladní software Conto [online]. 2010 [cit. 2013-05-04]. Dostupné z: http://www.consulta.cz/pokladni-software-conto/

18. TPOS. Oficiální stránky aplikace TPOS [online]. 2006 [cit. 2013-05-08]. Dostupné z: http://tpos.euweb.cz/

19. HANA, Kanisová a Miroslav MÜLLER. *UML srozumitelně*. Vyd. 1. Brno: Computer Press, 2004, 157 s. ISBN 80-251-0231-9.

20. A.W.I.S. SPRÁVA, SYSTÉMY S.R.O. *Pokladny a pokladní systémy* [online]. ©2013 [cit. 2013-03-04]. Dostupné z: http://www.kasa-pokladna.cz/

21. ORACLE CORPORATION. Oracle and Java. *Technologies* [online]. 2013 [cit. 2013-04-28]. Dostupné z: http://www.oracle.com/us/technologies/java/overview/index.html

22. Swing (Java), In: *Wikipedia: the free encyclopedia* [online]. St. Petersburg (Florida): Wikipedia Foundation, 11. 12. 2006, last modified 24.04.2013 [cit. 2013-04-29]. Dostupné z: http://en.wikipedia.org/wiki/Swing_%28Java%29

23. ORACLE CORPORATION. About MySQL. *MySQL* [online]. 2013 [cit. 2013-04-28]. Dostupné z: http://www.mysql.com/about/

24. MANAGEMENT MANIA. Třívrstvá architektura (Three-tier architecture). In: *Management Mania* [online]. 4.2.2013 [cit. 2013-04-08]. Dostupné z: https://managementmania.com/cs/trivrstva-architektura-three-tier-architecture

25. ORACLE CORPORATION. Enterprise JavaBeans Technology. *Oracle Corporation* [online]. [cit. 2013-04-29]. Dostupné z: http://www.oracle.com/technetwork/java/javaee/ejb/index.html

26. ORACLE CORPORATION. Vítejte u NetBeans. *NetBeans* [online]. 2013 [cit. 2013-04-28]. Dostupné z: https://netbeans.org/index_cs.html

27. ORACLE CORPORATION. Oracle GlassFish Server. *Oracle Corporation* [online]. 2011 [cit. 2013-04-29]. Dostupné z: http://www.oracle.com/us/products/middleware/application-server/050870.pdf

28. BERNARD, Borek. Úvod do architektury MVC. In: *Zdroják* [online]. 7. 5. 2009 [cit. 2013-04-29]. Dostupné z: http://www.zdrojak.cz/clanky/uvod-do-architektury-mvc/

29. MARCUSO, Sandro. MVC and Multi-tier architecture. In: *Crafted Software* [online]. 9. May 2010 [cit. 2013-04-29]. Dostupné z: http://craftedsw.blogspot.cz/2010/05/mvc-and-multi-tier-architecture.html

SEZNAM ZKRATEK

DSS	Decision support system (Systémy pro podporu rozhodování)
EIS	Executive information system (Systémy pro vrcholové řízení)
EJB	Enterprise JavaBeans
ERD	Entity-relationship diagram (Entitně-relační diagram)
FDD	Feature driven development (Vývoj řízený užitnými vlastnostmi)
GUI	Graphical user interface (Grafické uživatelské rozhraní)
IASW	Individuální aplikační software
ICT	Information and communication technologies (Informační a komunikační technologie)
IDE	Integrated development environment (Integrované vývojové prostředíú
IS	Information systém (Informační systém)
JFC	Java Foundation Classes
JVM	Java Virtual Machine
MIS	Management information system (Systémy pro řízení)
MSF	Microsoft solutions framework
MVC	Model-View-Controller architektura
PC	Personal computer (Osobní počítač)
RAD	Rapid application development (Rychlý vývoj aplikací)
RUP	Rational unified process
SQL	Structured query language (Strukturovaný dotazovací jazyk)
TASW	Typový aplikační software
TPS	Transaction processing system (Transakční systémy)
XP	Extreme programming (Extrémní programování)

Prohlášení o využití výsledků diplomové (bakalářské) práce

Prohlašuji, že

- jsem byl(a) seznámen(a) s tím, že na mou diplomovou (bakalářskou) práci se plně vztahuje zákon č. 121/2000 Sb. – autorský zákon, zejména § 35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a § 60 – školní dílo;
- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen VŠB-TUO) má právo nevýdělečně, ke své vnitřní potřebě, diplomovou (bakalářskou) práci užít (§ 35 odst. 3);
- souhlasím s tím, že diplomová (bakalářská) práce bude v elektronické podobě archivována v Ústřední knihovně VŠB-TUO a jeden výtisk bude uložen u vedoucího diplomové (bakalářské) práce. Souhlasím s tím, že bibliografické údaje o diplomové (bakalářské) práci budou zveřejněny v informačním systému VŠB-TUO;
- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona;
- bylo sjednáno, že užít své dílo, diplomovou (bakalářskou) práci, nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).

V Ostravě dne 10.05.2013

.....

Břetislav Glac

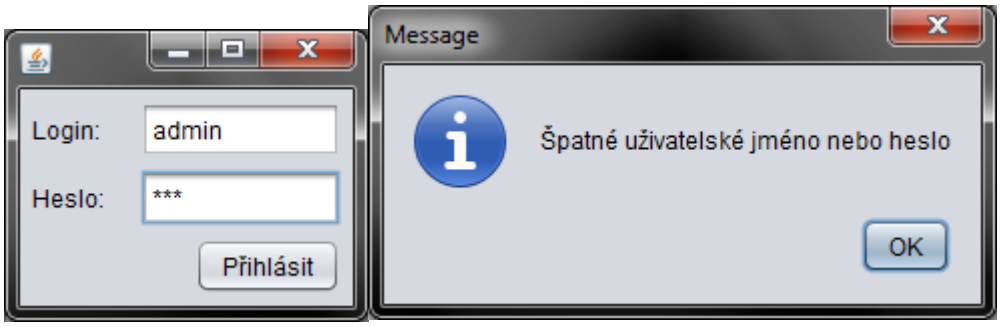
SEZNAM PŘÍLOH

Příloha č. 1: CD se zdrojovými kódy

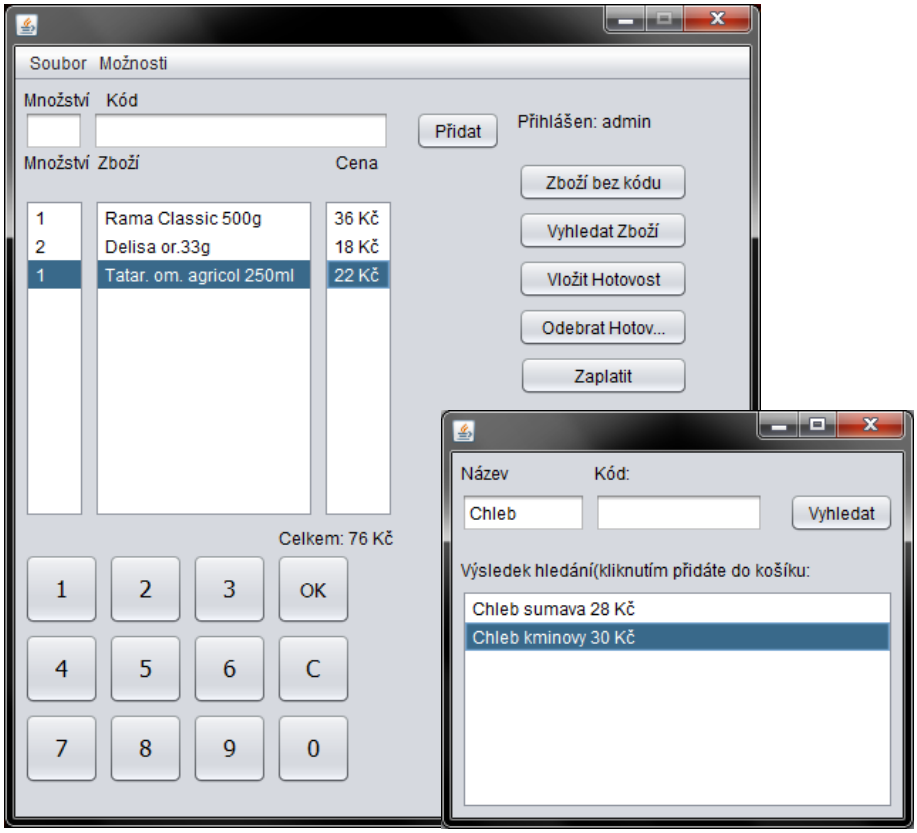
Na přiloženém CD se nachází:

- zdrojové kódy pro klientské aplikace.
- zdrojové kódy pro serverovou část systému.

Příloha č. 2: Ukázky grafického uživatelského rozhraní

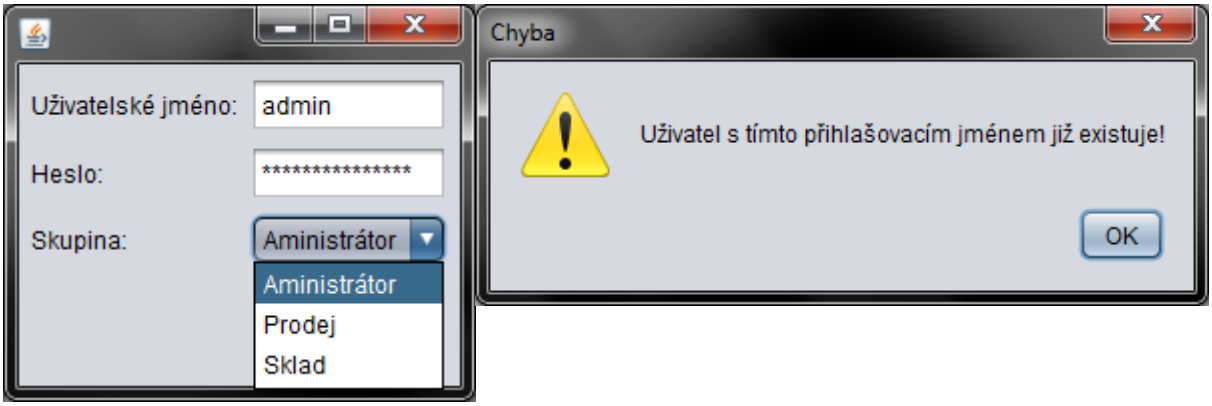


obr. 1: Ukázka přihlášení s chybovou hláškou

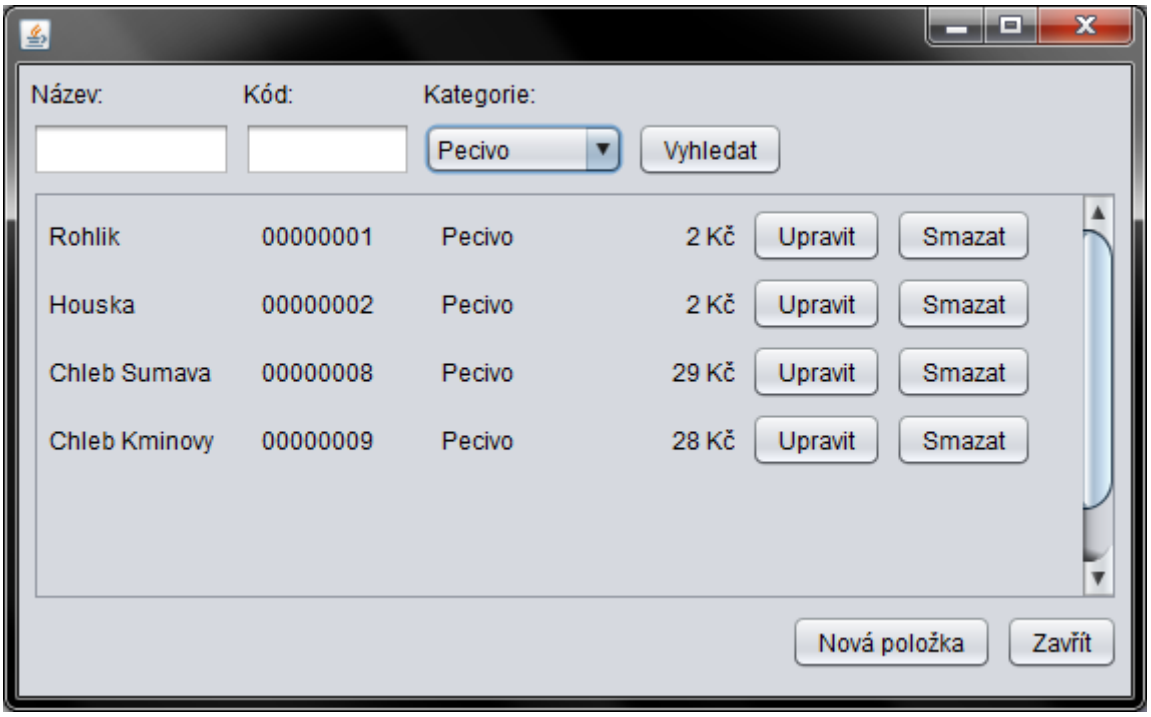


obr. 2: Hlavní obrazovka prodejní části s ukázkou vyhledávání

obr. 3: Skladová část - hlavní menu a obrazovka pro příjem zboží



obr. 4: Tvorba nového uživatele a chybová hláška



obr. 5: Obrazovka se seznamem a správou zboží